Abstracts from EuroCG 2011

27th European Workshop on Computational Geometry

Morschach, Switzerland
March 28-30, 2011

Preface

The 27th European Workshop on Computational Geometry (EuroCG 11) was held on March 28–30 at Antoniushaus Mattli in Morschach, Switzerland.

EuroCG is an annual, informal workshop whose goal is to provide a forum for scientists to meet, present their work, interact, and establish collaborations, in order to promote research in the field of Computational Geometry, within Europe and beyond.

We received 67 submissions, which underwent a limited refereeing process in order to ensure some minimal standards and to check for plausibility. We selected 56 submissions for presentation at the workshop, two of which were withdrawn later.

Apart from abstracts of the 54 contributed talks this booklet also contains abstracts of the three invited lectures, given by Mark de Berg, Friedrich Eisenbrand, and Jiří Matoušek. The content of this booklet—just like the pdf-files of abstracts available on the EuroCG website `http://eurocg.org`—should be regarded as a collection of preprints. Therefore, we expect most of the results presented here to be also submitted to peer-reviewed conferences and/or journals.

Many thanks to all authors and invited speakers for their participation, and to the members of the program committee and all external reviewers for their insightful comments. The nice coin on the title page (In CG we trust...) was designed by Yves Brise and he also deserves credit for borrowing me his camera to shoot the photos for the webpage. Finally, we are very grateful for the generous support of our sponsors: Choco Mundo GmbH, Sportbahnen Schwyz-Stoos-Fronalpstock AG, Disney Research Zürich, and ETH Zürich.

March 2011 Michael Hoffmann

Program Committee

- Bernd Gärtner, ETH Zürich
- Michael Hoffmann, ETH Zürich (chair)
- Gabriel Nivasch, ETH Zürich
- Shakhar Smorodinsky, Ben-Gurion University
- Bettina Speckmann, TU Eindhoven
- Marc van Kreveld, Utrecht University
- Emo Welzl, ETH Zürich

Organization

- Michael Hoffmann, ETH Zürich
- Andrea Salow, ETH Zürich
- Emo Welzl, ETH Zürich

External Reviewers

- Karim Abu-Afash
- Dominique Attali
- Gill Barequet
- Panagiotis Cheilaris
- Efi Fogel
- Stefan Funke
- Maarten Löffler
- Evanthia Papadopoulou
- Guodong Rong
- Oded Schwartz
- Tan Tiow Seng
- Tung Cao Thanh
- Xinyu Zhang
- Afra Zomorodian

Table of Contents

Session 3 (Tuesday 09:00–10:15)

Session 4 (Wednesday 10:30–12:00)

Session 5 (Wednesday 13:30–14:30)

Computational Geometry for Fat Objects and Low-Density Scenes

Mark de Berg

Department of Computer Science, TU Eindhoven
P.O. Box 513, 5600 MB Eindhoven, the Netherlands

In a traditional efficiency analysis in algorithms research, one studies the worst-case running time (or storage usage) of an algorithm as a function of the input size. In many cases such an analysis gives a good indication of the algorithm's efficiency in practice, because the worst-case behavior and the "normal" behavior are fairly similar. For geometric algorithms, however, this is often not the case: the worst-case behavior of a geometric algorithm may have little relation to its efficiency in practice. The main reason is that the worst-case behavior typically arises only for inputs with objects whose shapes or distribution over the space are quite extreme—much more extreme than occurs in practice. The conclusion is that expressing the efficiency of a geometric algorithm as a function of input size only is insufficient. What we need is a *geometry-sensitive analysis*, which also takes the shape and/or distribution of the input objects into account. In other words, we need to express the efficiency as a function of not only the input size, but also of certain parameters that capture the shape or distribution of the input.

This insight not only influences the way in which we should analyze geometric algorithms, it also has an impact on the design of algorithms: we need to design algorithms that become more efficient when the geometry-describing parameters have favorable values. Over the past 15 years or so, many such algorithms have been developed. Most of these either use the *fatness* of the input objects as a parameter, or their *density*. The former parameter says something about the shape of the individual objects, while the latter is concerned with their distribution over the space. In this talk I will give an overview of the results that have been obtained in this area.

Recent Trends in Algorithmic Geometry of Numbers

Friedrich Eisenbrand*

Abstract

The algorithmic geometry of numbers is a classical field of mathematics that was pioneered by Lagrange and Gauss with their work on binary quadratic forms more than 200 years ago. Questions concerning the representability of integers by quadratic forms and geometric techniques to answer them have fascinated mathematicians since then.

At least since the development of the LLL-algorithm more than 25 years ago, the algorithmic geometry of numbers has played an important role in the fields of computational complexity, cryptography, and optimization. Here, in particular, fundamental lattice problems like closest and shortest vectors of a lattice play a prominent role.

In this tutorial I will survey some results from the algorithmic geometry of numbers and its relation to algorithmic research. I plan to start with exemplary results from the origins of the field and build a bridge to recent algorithmic breakthroughs and intriguing open problems.

*EPFL, Station 8, 1015 Lausanne, friedrich.eisenbrand@epfl.ch

The Dawn of an Algebraic Era in Discrete Geometry?

Jiří Matoušek*

Section One, Where the Author Is Browsing the 2010 ArXiv

To me, 2010 looks as *annus mirabilis*, a miraculous year, in several areas of my mathematical interests. Below I list seven highlights and breakthroughs, mostly in discrete geometry, hoping to share some of my wonder and pleasure with the readers.

Of course, hardly any of these great results have come out of the blue: usually the paper I refer to adds the last step to earlier ideas. Since this is an extended abstract (of a nonexistent paper), I will be rather brief, or sometimes completely silent, about the history, with apologies to the unmentioned giants on whose shoulders the authors I do mention have been standing.[1] A careful reader may notice that together with these great results, I will also advertise some smaller results of mine.

- Larry Guth and Nets Hawk Katz [16] completed a bold project of György Elekes (whose previous stage is reported in [10]) and obtained a near-tight bound for the **Erdős distinct distances problem**: they proved that every n points in the plane determine at least $\Omega(n/\log n)$ distinct distances. This almost matches the best known upper bound of $O(n/\sqrt{\log n})$, attained for the $\sqrt{n} \times \sqrt{n}$ grid. Their proof and some related results and methods constitute the main topic of this note, and will be discussed later.

- János Pach and Gábor Tardos [27] found tight lower bounds for the **size of ε-nets** for geometric set systems.[2] It has been known for a long time that for systems such as halfspaces, balls, simplices in $\mathbb{R}^d$, for d fixed, ε-nets of size $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ exist, for every finite set X and every ε. The proof was based on a very general combinatorial property of the considered set system, called *bounded VC-dimension*, and there was hope that simple geometrically defined systems might admit even smaller ε-nets, perhaps of size $O(\frac{1}{\varepsilon})$. Indeed, there were some reasons for optimism, since $O(\frac{1}{\varepsilon})$ was known for halfspaces in $\mathbb{R}^3$, and more recently, Aronov, Ezra, and Sharir [2] proved an $O(\frac{1}{\varepsilon}\log\log\frac{1}{\varepsilon})$ upper bound for axis-parallel rectangles in $\mathbb{R}^2$. However, Alon [1] established the first superlinear lower bound in a geometric setting (for lines in the plane), and Pach and Tardos got the tight lower bound of $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ for halfspaces in $\mathbb{R}^4$, as well as $\Omega(\frac{1}{\varepsilon}\log\log\frac{1}{\varepsilon})$ for axis-parallel rectangles in $\mathbb{R}^2$. These results may perhaps not look as significant to others, but for me, they close a long open and tantalizing problem, which for myself I considered almost hopeless. Moreover, I think that the proofs contain (reinforce?) a general lesson: in order to prove an "irregularity" result, in the sense that something cannot be very uniform, it may often be good to strive for a Ramsey-type result, showing that there has to be a completely non-uniform, "monochromatic" spot—in the case of ε-nets, this way even gives a tight quantitative bound!

- Mikhail Gromov [14] invented a new **topological proof of the first selection lemma**. The lemma states that for every n-point set $P \subset \mathbb{R}^d$ there exists a point a (not necessarily in P) contained in at least $c_d\binom{n}{d+1}$ of the d-dimensional simplices with vertices in P, where c_d is a positive constant depending only on d. (There are $\binom{n}{d+1}$ d-simplices spanned by P in total, so a is in a positive fraction of them.) Gromov's proof yields significantly better value of c_d than all previous ones, provides a far-reaching generalization, and hopefully opens a way towards deeper understanding of many related problems. By including Gromov's paper in my 2010 list I am cheating slightly, since a preprint was circulated one or two years earlier. But a completely honest 2010 item was supplied by Karasev [21], who found a greatly simplified and fairly elementary version of the argument. Readers interested in the com-

*Department of Applied Mathematics and Institute of Theoretical Computer Science (ITI), Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic, and Institute of Theoretical Computer Science, ETH Zurich, 8092 Zurich, Switzerland

[1] I should also add that my selection is entirely personal and subjective, shall not indicate or imply any ranking of the results listed, any judgment of results not mentioned, or any comparison thereof, and shall not encompass any warranty of sound and safe conditions of the cited papers or any liability on my side, in particular, no rights of readers or third parties to be indemnified in case of loss or damage directly or indirectly related to the aforementioned papers. But I do not want this abstract to look like a car rental contract.

[2] Let $\mathcal{F}$ be some system of subsets of $\mathbb{R}^d$, say all closed halfspaces or all axis-parallel boxes, and let X be a finite set in $\mathbb{R}^d$. A subset $N \subseteq X$ is called an *ε-net* for X with respect to $\mathcal{F}$, where $\varepsilon \in [0,1]$ is a given real number, if N intersects every "large" set, i.e., every $F \in \mathcal{F}$ for which $|F \cap X| \geq \varepsilon|X|$.

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

binatorial issues involved and in attempts at explaining Gromov's method may also reach for a paper of Wagner and mine [25].

- Francisco Santos [28] disproved the 1957 **Hirsch conjecture**, which states that the graph of a d-dimensional convex polytope with n facets has diameter at most $n-d$. The conjecture was motivated by linear programming, more precisely, by the analysis of simplex-type algorithms. Santos' ingenious examples have diameter $(1+\varepsilon)(n-d)$ for a small positive ε, and the fascinating question of maximum diameter of a d-polytope with n facets has become even more interesting (and the subject of Gil Kalai's polymath project), the best upper bound being $n^{O(\log d)}$ [18].

- Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick [13] proved very strong **lower bounds for various randomized simplex algorithms**.

  The simplex method from the late 1940s remains one of the best linear programming algorithms in practice, but a construction known as the *Klee–Minty cube* showed in the 1970s that some of the widely used variants are exponential in the worst case. This started a quest for a polynomial-time version, and on the optimistic side, a subexponential upper bound, of roughly $\exp(O(\sqrt{n}))$, was proved in 1992 for an algorithm known as RANDOM FACET. There was hope that the analysis might be further improved, or that some other randomized variant could be shown to be polynomial.

  Friedmann et al. shattered these great expectations almost completely, by proving an almost matching lower bound for RANDOM FACET, as well as a similar lower bound for another promising-looking algorithm known as RANDOM EDGE (lower bounds of this kind were known before, but only for the performance of these algorithms on certain "generalized linear programs", while the possibility of polynomial bounds for actual linear programs was still open). They use a powerful new way of constructing "difficult" linear programs, based on randomized *parity games*. The potential of this approach apparently has not yet been exhausted.

- Nikhil Bansal [3] found a polynomial-time **algorithm for computing low-discrepancy colorings**,[3] using semidefinite programming and ingenious rounding via a high-dimensional random walk. The algorithm is not an approximation algorithm for discrepancy in general (indeed, approximating discrepancy is pretty much hopeless [5]), but it makes several existential bounds for the discrepancy of certain set systems, such as all arithmetic progressions on $\{1, 2, \ldots, n\}$, constructive, which has been a major open problem in discrepancy theory. It also has structural consequences; one of the quick spinoffs is a near-tight answer [24] to an old questions of Sós concerning the discrepancy of a union of two set systems on the same ground set (besides Bansal's algorithm, the answer also relies on a beautiful linear-algebraic lower bound for discrepancy of Lovász, Spencer, and Vesztergombi [22]).

- June Huh [17] proved **log-concavity of the sequence of coefficients of the chromatic polynomial**.[4] More precisely, for the chromatic polynomial written as $a_0+a_1x+\cdots+a_nx^n$, Huh's result asserts that $a_{i-1}a_{i+1} \le a_i^2$ for every i (and for an arbitrary graph G, of course). This implies that the sequence $(|a_0|, |a_1|, \ldots, |a_n|)$ is *unimodal*, a 1968 conjecture of Read.

  The proof relies on connections of the problem to singularities of local analytic functions and ultimately to mixed multiplicities of certain ideals. This result does not entirely fit my list since it cannot be passed for discrete geometry even with considerable indulgence, but it looks beautiful and it does rest on geometric ideas. I do not understand much of it, but perhaps some day I will have enough time and energy to learn the necessary background or someone will explain it to me—at least it does not look as intimidating as some other papers.

Section Two, On Distinct Distances and Other Algebraic Magic

The following three problems were raised by Erdős [12] in 1946:

- Estimate the maximum possible number of **incidences** between a set P of m points and a set L of n lines in the plane (where an incidence is a pair (p, ℓ) with $p \in P$, $\ell \in L$, and $p \in \ell$).

- Estimate the maximum number of **unit distances** among n points in the plane. This can also be reformulated using incidences; up to a multiplicative factor of at most 2, one wants to estimate the maximum number of incidences between n unit circles and n points in the plane.

[3] The input to the algorithm is a set system $\mathcal{F}$ on a finite set X, and the output is a coloring χ of X by $+1$s and -1s. The *discrepancy* $\operatorname{disc}(\chi, \mathcal{F})$ of χ is the maximum, over all sets $F \in \mathcal{F}$, of the "imbalance" of F, i.e., of $\left|\sum_{x \in F} \chi(x)\right|$. The discrepancy of $\mathcal{F}$ is $\min_\chi \operatorname{disc}(\chi, \mathcal{F})$, where the minimum is over all possible ± 1 colorings of X.

[4] The *chromatic polynomial* of a graph G is a polynomial whose value at a natural number k equals the number of proper coloring of G with k colors. It can be shown that such a polynomial indeed exists, and is determined uniquely, for every G.

- Estimate the minimum number of **distinct distances** determined by n points in the plane.

The first two of these problems, incidences and unit distances, look quite similar at first sight. For point-line incidences the order of magnitude was determined precisely by Szemerédi and Trotter [31] in 1983; in particular, for n points and n lines the bound is of order $n^{4/3}$. Two other, much simpler proofs were discovered later, by Clarkson et al. [6] and by Székely [30]. These proofs are "combinatorial", in the sense that they do not use the straightness of the lines (and thus they also work for incidences of points with *pseudolines*, i.e., with systems of curves satisfying certain simple combinatorial axioms).

For unit distances, or incidences of unit circles with points, a modification of these proofs also yields an $O(n^{4/3})$ upper bound, but here a matching lower bound is lacking—the best known example, a suitable grid, yields only a near-linear bound, smaller than $n^{1+\delta}$ for every fixed $\delta > 0$. Erdős conjectured this lower bound to be essentially optimal.

Many ingenious techniques have been developed for bounding the number of incidences in various settings; see, e.g., a recent survey by Pach and Sharir [26]. However, none of them seems capable of breaking the $n^{4/3}$ barrier. Moreover, the unit distance problem can be considered not only for the Euclidean norm, but also for other norms in the plane, and Valtr [32] constructed a norm for which n-point sets actually exist with $\Omega(n^{4/3})$ unit distances. This indicates that limits of combinatorial approaches to the unit distance problem, powerful as they are, have been reached.

It seems that for further progress on unit distances (in the upper bound, which should be the way to go according to Erdős and general belief) one has to use some algebraic properties of the circle.

Indeed, while the unit distances problem still stands, the distinct distances problem was essentially settled by Guth and Katz, as was already mentioned—and their proof combines ingenious algebraic arguments (using tools from algebraic geometry, mostly dating back to the 19th century) with geometric, or perhaps topological, considerations similar to those appearing in earlier higher-dimensional incidence proofs.

The Guth–Katz proof is somewhat complicated (and I am sure simpler variants will be found sooner or later), and I will not attempt to even sketch it here. I will just mention two of its main ingredients.

One of them is a simple but surprisingly powerful trick invented by Dvir [7], which can be illustrated on another remarkable recent achievement in incidence problems, the *joints problem* (solved in another paper of Guth and Katz [15], with further simplifications and extensions by Elekes, Kaplan, Sharir, Shustin, and Quillodrán; the following outline mostly follows [19]). We consider a set L of n lines in $\mathbb{R}^3$, and call a point $a \in \mathbb{R}^3$ a *joint* if there are at least three lines of L, not all coplanar, passing through a. The question is, what is the maximum possible number of joints for n lines, and the answer is $O(n^{3/2})$ (with a matching lower bound example provided by a suitable grid of lines). The proof goes as follows.

For contradiction, we suppose that a set L of n lines has at least $Cn^{3/2}$ joints, with C very large (and n even much larger). Let J be the set of all joints of L, and let $m := |J|$.

We first need to prune L and J so that all of the remaining lines contain many joints. By a simple pruning procedure (repeatedly discarding lines with a small number of joints and all the joints incident to them) we can select $L' \subseteq L$ and $J' \subseteq J$ so that every $\ell \in L'$ contains at least $k := m/2n$ points of J', and each point of J' is a joint of the lines in L'.

Next, a simple argument shows that there exists a nonzero polynomial in three variables that vanishes on all points of J' and has degree at most $4m^{1/3}$ (the condition of vanishing on J' is expressed by a system of homogeneous linear equations with the coefficients of the desired polynomial as unknowns, and when the number of unknowns exceeds the number of equations, a nonzero solution exists). Among all nonzero polynomials vanishing on J' we choose one of the smallest possible degree and call it f; we thus have $D := \deg(f) \le 4m^{1/3}$.

Let us consider the restriction f_ℓ of f to a line $\ell \in L'$. It vanishes at each of the at least k points of J' incident to ℓ. Since f_ℓ can be regarded as a univariate polynomial of degree at most D, either it is 0 everywhere on ℓ, or it has at most D zeros there. But for large C we have $D < k$, and so we get that $f_\ell \equiv 0$. Thus f vanishes on all lines of L'.

Now we let $\vec{g} := \nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$ be the gradient of f (this is a polynomial mapping $\mathbb{R}^3 \to \mathbb{R}^3$). Since f vanishes along every $\ell \in L'$, the projection of $\vec{g}(p)$ onto ℓ also vanishes for all $p \in \ell$. In other words, $\vec{g}(p)$ is perpendicular to ℓ. In particular, if we look at a joint $a \in J'$, we obtain that $\vec{g}(a)$ is perpendicular to three non-coplanar lines, and hence $\vec{g}(a) = (0, 0, 0)$.

Since taking a partial derivative decreases the degree of a polynomial, each of the three components of $\vec{g}$ is a polynomial of degree *smaller* than D, and we have just proved that it vanishes on J'. Since f was the smallest-degree nonzero polynomial vanishing on J', $\vec{g}$ must be identically 0. This means than f is constant, and this is a contradiction (since we assumed it to be nonzero and to vanish on J'), which concludes the proof.

This was Dvir's trick in action, and now, what *is* the trick? An idea turned into a recipe often loses much of its usefulness, but a vague general formulation might perhaps be this: one constructs a low-

degree polynomial vanishing on the considered points, or other "low-dimensional" objects, infers that it has to vanish on some other objects of higher dimension as well, and derives a global conclusion about the polynomial, such as vanishing of the gradient.

Another ingredient of the distinct distances proof, which was invented by Guth and Katz for this purpose, are *space partitions using polynomials*. Given a finite set $P \subset \mathbb{R}^d$ and a parameter r, $1 < r \leq |P|$, they apply the *polynomial ham-sandwich theorem* of Stone and Tukey to obtain a partition of $\mathbb{R}^d$. Namely, they construct a nonzero polynomial f so that no connected component of $\mathbb{R}^d \setminus Z(f)$ contains more than $|P|/r$ points of P (where $Z(f)$ is the zero set of f). A key feature of the construction is that $\deg(f)$ need not be too high, only $O(r^{1/d})$, and thus the interaction of other objects, such as lines or hyperplanes, with $Z(f)$ is under control in some sense. This method provides an alternative to previous space partitioning techniques, such as *cuttings* or *simplicial partitions*, and in some respects it is apparently more powerful than these earlier tools.

A simple example of using these polynomial partitions is a new simple proof of the Szemerédi–Trotter upper bound on point-line incidences. Together with some other applications, we have described this in detail in a recent preprint with Kaplan and Sharir [20], and so it will not be discussed here.

There are several other encouraging examples of "enriching discrete geometry with algebra" besides those mentioned above, a major share of them due to Elekes and his co-authors.

One of the major directions are *sum-product theorems*, originally a number-theoretic issue. The earlier stages, dealing with real numbers, are nicely surveyed in Elekes [8]. Then, after Bourgain, Katz, and Tao [4] obtained a sum-product theorem in finite fields, there has been a great surge of activity, and the sum-product business has been extended far beyond the borders of discrete geometry (and far beyond the intended scope of the present note). It has found many diverse applications, including in number theory, group theory, and the explicit constructions of extractors and Ramsey graphs. An accessible initial picture and references can be gained, e.g., from a talk of Wigderson as recorded in the lecture notes at `http://www.math.cmu.edu/~af1p/Teaching/AdditiveCombinatorics/allnotes.pdf`.

Another interesting (and related) direction can be represented, e.g., by the papers of Elekes and Rónyai [9] and of Elekes, Simonovits, and Szabó [11]. I will mention only a rather concrete consequence of the results of [9], known as *Purdy's conjecture*. Let C be a constant, let a and b be lines in the plane, let $p_1, \ldots, p_n$ be distinct points on a, and let $q_1, \ldots, q_n$ be distinct points on b. Purdy's conjecture, now a theorem, states that if the set of the Euclidean distances $\{\|p_i - q_j\| : i, j = 1, 2, \ldots, n\}$ has at most Cn distinct elements, and if n is sufficiently large, then a and b must be either parallel or perpendicular. Actually, it seems that it should be sufficient to assume at most $n^{2-\varepsilon}$ distinct distances, for some fixed $\varepsilon > 0$, and the conclusion should still hold, but this is wide open at present.

The last two "algebraic" contributions mentioned here concern the unit distances problem, or rather, they deal with variants of it. Schwartz et al. [29] proved that if we count only unit distances with *rational angles* (i.e., attained for pairs of points p, q such that the angle of the line pq with the x-axis is a rational multiple of π), then the number of these special unit distances is bounded by $O(n^{1+\varepsilon})$, for every fixed $\varepsilon > 0$. In [23], I proved that there *exist* norms in the plane for which the number of unit distances obeys the upper bound $O(n \log n \log\log n)$ (while *every* norm admits sets with $\Omega(n \log n)$ unit distances, and, according to Valtr [32], *some* norms allow for as many as $\Omega(n^{4/3})$ unit distances). The proof combines a linear-algebraic argument, a graph-theoretic lemma we proved earlier for another purpose with Přívětivý and Škovroň, and a Baire category argument.

The above short survey certainly does not exhaust all significant examples of algebraic methods used in discrete geometry. Moreover, I have no doubts that, while I am writing this, bright mathematicians are working hard on extending the Guth–Katz breakthrough and on other amazing algebraic methods and applications.

So, does this signify the beginning of an algebraic era in discrete geometry? My list at the beginning of this note shows a more balanced picture, with topology, probability, optimization, and ingenuity in a friendly competition with algebra, but it indicates that the best results have usually been achieved not only by cleverness and new ideas, but also by applying more and more advanced tools from various branches of mathematics. So I am not sure about an algebraic era, but, just in case, recently I have ordered five textbooks on algebraic geometry and such things for my PhD students and myself.

Acknowledgment. I would like to thank Micha Sharir, Imre Bárány, and Uli Wagner for valuable comments to a draft of this note.

References

[1] N. Alon: A non-linear lower bound for planar epsilon-nets, in *Proc. 51st Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)* 2010, pages 341–346.

[2] B. Aronov, E. Ezra and M. Sharir: Small-size epsilon-nets for axis-parallel rectangles and boxes, *SIAM J. Comput.* 39(2010), 3248–3282.

[3] N. Bansal: Constructive algorithms for discrepancy minimization, in *Proc. 51st IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010. Full version arXiv:1002.2259.

[4] J. Bourgain, N. H. Katz and T. C. Tao: A sum-product estimate in finite fields, and applications, *Geom. Funct. Anal.* 14(2004) 27–57.

[5] M. Charikar, A. Newman, and A. Nikolov: Tight hardness results for minimizing discrepancy. In *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

[6] K. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir and E. Welzl, Combinatorial complexity bounds for arrangements of curves and spheres, *Discrete Comput. Geom.* 5 (1990), 99–160.

[7] Z. Dvir: On the size of Kakeya sets in finite fields, *J. Amer. Math. Soc.* 22(2009) 1093–1097.

[8] Gy. Elekes, Sums versus products in number theory, algebra and Erdős geometry, in G. Halász et al. editors, *Paul Erdős and His Mathematics*, J. Bolyai Math. Soc., Budapest, 2002, pages 241–290.

[9] Gy. Elekes, L. Rónyai: A combinatorial problem on polynomials and rational functions, *J. Comb. Theory, Ser. A* 89(2000) 1–20.

[10] Gy. Elekes and M. Sharir, Incidences in three dimensions and distinct distances in the plane, in *Proc. 26th Annu. ACM Sympos. Comput. Geom.*, pages 413–422, 2010.

[11] Gy. Elekes, M. Simonovits, E. Szabó: A combinatorial distinction between unit circles and straight lines: How many coincidences can they have? *Combinat. Probab. Comput.* 18(2009) 691–705.

[12] P. Erdős, On a set of distances of n points, *Amer. Math. Monthly* 53 (1946), 248–250.

[13] O. Friedmann, T. Hansen, U. Zwick: Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proc. 43rd ACM Symposium on Theory of Computing (STOC)*, 2011, to appear.

[14] M. Gromov: Singularities, expanders and topology of maps. Part 2: From combinatorics to topology via algebraic isoperimetry. *Geom. Funct. Anal.*, 20(2010) 416–526.

[15] L. Guth and N. H. Katz: Algebraic methods in discrete analogs of the Kakeya problem, *Adv. Math.* 225(2010) 2828–2839.

[16] L. Guth and N. H. Katz, On the Erdős distinct distances problem in the plane, preprint, arXiv:1011.4105.

[17] J. Huh: Milnor numbers of projective hypersurfaces and the chromatic polynomial of graphs, preprint, arXiv:1008.4749.

[18] G. Kalai and D. J. Kleitman: A quasi-polynomial bound for the diameter of graphs of polyhedra, *Bull. Amer. Math. Soc.* 26(1992) 315–316.

[19] H. Kaplan, M. Sharir, and E. Shustin: On lines and joints, *Discrete Comput. Geom.* 44(2010) 838–843.

[20] H. Kaplan, J. Matoušek, and M. Sharir: Simple proofs of classical theorems in discrete geometry via the Guth–Katz polynomial partitioning technique, preprint, arXiv (number not yet available at the time of writing).

[21] R. Karasev: A simpler proof of the Boros–Füredi–Bárány–Pach–Gromov theorem, preprint, arXiv:1012.5890, 2010.

[22] L. Lovász, J. Spencer, and K. Vesztergombi: Discrepancy of set-systems and matrices, *European J. Combinat.*, 7(1986) 151–160.

[23] J. Matoušek: The number of unit distances is almost linear for most norms, *Adv. Math.* 226(2011) 2618–2628. Preprint arXiv:1007.1095.

[24] J. Matoušek: The determinant bound for discrepancy is almost tight, preprint, arXiv:1101.0767.

[25] J. Matoušek, U. Wagner: On Gromov's method of selecting heavily covered points, preprint, arXiv:1102.3515.

[26] J. Pach and M. Sharir: Geometric incidences, in *Towards a Theory of Geometric Graphs* (J. Pach, ed.), *Contemporary Mathematics*, Vol. 342, Amer. Math. Soc., Providence, RI, 2004, pages 185–223.

[27] J. Pach, G. Tardos: Tight lower bounds for the size of epsilon-nets, In *Proc. 27th ACM Sympos. Comput. Geom.*, 2011, to appear; also arXiv:1012.1240.

[28] F. Santos: A counterexample to the Hirsch conjecture, preprint, arXiv:1006.2814.

[29] R. Schwartz, J. Solymosi, and F. de Zeeuw: Rational distances with rational angles, preprint, arXiv:1008.3671.

[30] L. Székely: Crossing numbers and hard Erdős problems in discrete geometry, *Combinat. Probab. Comput.* 6(1997) 353–358.

[31] E. Szemerédi and W. T. Trotter: Extremal problems in discrete geometry, *Combinatorica* 3(1983) 381–392.

[32] P. Valtr, Strictly convex norms allowing many unit distances and related touching questions, manuscript, Charles University, Prague 2005.

Constructing Optimal Shortcuts in Directed Weighted Paths

R. Klein* M. Krug† E. Langetepe* D. T. Lee‡ D. Wagner†

Abstract

We show that the optimum shortcut, with respect to routing cost, in a directed path with weighted vertices can be found in linear time. To this end, we construct in linear time the lower envelope of an arrangement of pseudo-lines whose order at infinity is given. The algorithm can also be applied to star-shaped directed networks.

Keywords: Arrangements, computational geometry, lower envelope, network optimization, pseudoline, routing cost, shortcut edge.

1 Introduction

Transportation networks can be modelled by graphs in a number of different ways. Quite frequently, vertices represent cities, and edges stand for highway or railroad connections between them. In the *routing cost* model, the overall quality of a network is measured by the sum of the lenghts of all shortest paths between all pairs of vertices; see Wu and Chao [6]. Here the length of a path equals the sum of the lengths of its edges.

We are interested in augmenting a given network, by adding an extra edge to it in such a way that its routing cost is reduced as much as possible. In our model each edge is of unit length, reflecting a situation where long distance traffic is fast, while changing between highways or railways requires time consuming inner-city travel.

This augmenting problem has been studied by Farshi et al. [2] with respect to the dilation (or: stretch factor, spanning ratio) of a network. Their problem differs considerably from ours. For example, the geometric path P over n vertices shown in Figure 1 is of dilation $2/\epsilon$, attained by vertices v_1 and v_3, while its routing cost

$$2\sum_{1\le i<j\le n}(j-i)=\frac{(n-1)n(n+1)}{6}$$

does not depend on the geometric embedding of P. With one extra edge available, we would minimize the dilation of P by inserting it between v_1 and v_3, whereas the routing cost of P is minimized by adding an edge connecting $v := v_{n/5}$ to $v' := v_{4n/5}$; the proof of optimality requires lenghty calculations. Not only the paths between vertices to the left of v and to the right of v' become shorter; vertices between v and v' also benefit from the shortcut edge, as Figure 1 indicates.

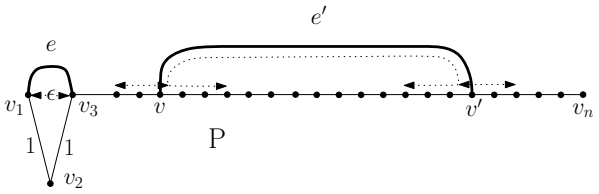


Figure 1: Inserting edge e minimizes the dilation, while e' minimizes the routing cost of path P.

In this note we consider the following problem. We are given a path $P = (v_1, v_2, \ldots, v_n)$ all of whose edges (v_i, v_{i+1}) are directed from left to right. Each vertex v_i is assigned a weight $w_i > 0$, reflecting the number of residents of city v_i. We want to decrease the routing cost

$$r := \sum_{1\le i<j\le n} w_i(j-i)w_j$$

of P by adding one more edge (v_k, v_l) directed from v_k to v_l, choosing $k < l$ such that the decrease in routing cost is as large as possible; compare Figure 2.

As compared to the undirected case shown in Figure 1, the situation has become simpler in that only vertex pairs on opposite sides of the shortcut edge can benefit from the extra edge. Thus, the decrease in routing cost effected by adding a directed edge from v_k to v_l equals

$$\rho(k,l) := \left(\sum_{1\le i\le k} w_i\right)(l-k-1)\left(\sum_{l\le i\le n} w_i\right), \quad (1)$$

because $l-k$ edges between v_k and v_l can be avoided by using edge (v_k, v_l).

On the other hand, introducing weights has made the minimization problem non-trivial. In fact, the decrease function $\rho(k,l)$ can have multiple local maxima. (An example is given by the path with weights $(90, 10, 100, 1, 100, 1, 1, 1, 100, 1, 100, 10, 90)$ for which the shortcut edges (v_3, v_{11}) and (v_5, v_9) are locally optimal.) Quite obviously, the optimum shortcut edge could be determined by inspecting the $O(n^2)$ many candidates (v_k, v_l) where $1 \le k < l \le n$.

*Department of Computer Science I, University of Bonn
†Karlsruhe Institute of Technology (KIT), Institute for Theoretical Informatics
‡Institute of Information Science, Academia Sinica

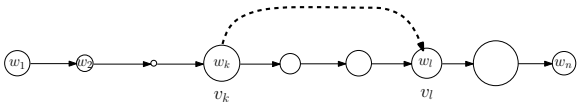


Figure 2: Weights w_i represent the population size of city v_i.

In this note, we are going to prove a sharper result.

Theorem 1 *The optimum shortcut edge for a directed, weighted path of n vertices can be determined in time $O(n)$.*

2 Reduction to Pseudo-Line Arrangements

Let P denote a directed path of n vertices $v_1, \dots v_n$ with positive weights $w_1, \dots, w_n$. Considering Formula 1 we define

$$A_k := \sum_{i=1}^{k} w_i \quad \text{and} \quad B_l := \sum_{i=l}^{n} w_i$$

for $k = 1, \dots, n$. Moreover, let

$$f_l(k) := A_{\lfloor k \rfloor}(l - k - 1)B_l$$

be a function of a real variable k. In our optimization problem, only values $k < l$ are meaningful, but there is no harm in ignoring this constraint. Function $f_l(k)$ is piecewise linear, with discontinuities at integer values of k. Here, $f_l(k) = \rho(k, l)$ holds.

Lemma 2 *Let $1 \le l_1 < l_2 \le n$ be fixed.*
(i) There exists at most one real value $k \in [1, n]$ where $f_{l_1}(k) = f_{l_2}(k)$ holds. If so, we have $f_{l_1}(k) < f_{l_2}(k)$ to the right of k.
(ii) If no such value exist, $f_{l_1}(k) < f_{l_2}(k)$ holds over the whole interval $[1, n]$,

Proof. Let us consider

$$f_{l_2}(k) - f_{l_1}(k) =$$
$$A_{\lfloor k \rfloor} \cdot (\ (l_2 - 1)B_{l_2} - (l_1 - 1)B_{l_1} + k(B_{l_1} - B_{l_2})\ ). \quad (2)$$

Because of $l_1 < l_2$ we have $B_{l_1} > B_{l_2}$. Moreover, $A_{\lfloor k \rfloor} \le A_{\lfloor k' \rfloor}$ holds if $k < k'$. Thus, Expression 2 is strictly monotonically increasing in k, which proves Claim (i). If Expression 2 never attains the value 0, then the order relation between $f_{l_1}(k)$ and $f_{l_2}(k)$ is the same all over $[1, n]$. If we set $k := l_2 - 1 \in [1, n]$ we obtain $f_{l_1}(k) < 0 = f_{l_2}(k)$, which completes the proof of Claim (ii). □

The set of graphs of functions $f_l(k)$, where $1 \le l \le n$, is called a family of *pseudo-lines* over the interval $[1, n]$ because any two of them have at most one point of intersection, just as proper lines would. Arrangements of pseudo-lines have been extensively studied; see Goodman [4], for example. We can solve our optimization problem by constructing the *upper envelope* of this pseudo-line arrangement, that is, the graph G of the maximum function

$$f(k) := \max\{f_l(k); 1 \le l \le n\}. \quad (3)$$

Each function $f_l(k)$ contributes at most one segment to G. Indeed, let us assume that some $f_l(k)$ contributed two segments to G. Then a segment of some $f_{l'}(k)$, where $l' \neq l$, must occur in between. But for this to happen, $f_l(k)$ and $f_{l'}(k)$ must intersect twice—in contradiction to Lemma 2.

This is a special, and in fact the most simple, case of a Davenport-Schinzel sequence. In general, if any two of n function graphs over some interval intersect at most s times, their envelope is of complexity $O(\lambda_s(n))$, with a non-trivial, slightly super-linear function λ_s; see the monograph by Sharir and Agarwal [5]. There is a simple algorithm that allows the lower (or upper) envelope to be constructed in time $O(\lambda_s(n) \log n)$, by divide-and-conquer. Here one assumes that elementary operations, like computing an intersection of two functions, can be carried out in constant time. In our case, this is true. Moreover, $s = 1$ and $\lambda_1(n) = n$ hold.

These facts give us a first improvement over the trivial $O(n^2)$ algorithm mentioned in Section 1. We can construct the upper envelope, G, in time $O(n \log n)$. Then we perform one pass over G and evaluate $f(k)$ at all integer values of k; this takes time $O(n)$. If the maximum of these values is attained within a segment of $f_l(k)$ in G, then the shortcut edge from v_k to v_l yields a maximum reduction in routing cost.

3 Computing the Envelope in Linear Time

To improve on the $O(n \log n)$ upper time bound just mentioned, we will make use of the fact that the ordering of the functions $f_l(k)$ "far to the right" is known to us. Indeed, for each $l \le n - 1$, Expression 2 yields

$$f_{l+1}(n) - f_l(n) = A_n(B_{l+1} + (n + 1 - l)w_l) > 0,$$

so that we obtain

$$f_1(n) < f_2(n) < \ldots < f_n(n).$$

We prove a general result covering this situation.

Theorem 3 *Let $f_1, \dots, f_n$ form an arrangement A of pseudo-lines over interval $I = [a, b]$. If the order of values $f_i(b)$ is known, the upper envelope of A can be computed in time $O(n)$.*

Proof. We proceed from right to left and process the curves f_j in order of decreasing indices. In a stack, S, we store the part of the upper envelope that would result if no further curves existed; see Figure 3. Initially, S contains only f_n.

When processing f_j, we compute the intersection, w, of f_j with the top element, f_k, of S. If w does not exist, or lies to the left of interval $I = [a, b]$, we ignore f_j, and start processing f_{j-1}.

Otherwise, let v be the intersection of the two topmost curves in S. If w lies to the left of v, or if no v exists because the stack contains only one element, $f_k = f_n$, we push f_j on the stack, and start processing f_{j-1}. But if w lies to the right of v, we pop f_k from the stack and continue processing f_j. This pop

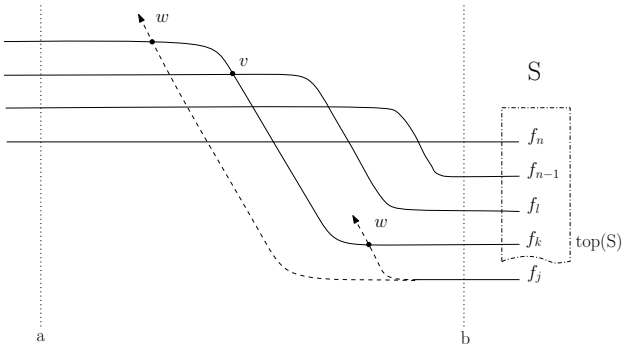


Figure 3: Constructing the upper envelope.

operation is justified because to the left of w, curve f_k is dominated by f_j and can, therefore, not contribute to the upper envelope. With this observation, the correctness of our algorithm is evident.

The linear run time bound can be shown as follows. During each pass through the loop described above, we (i) ignore f_j, or (ii) push f_j, or (iii) we pop f_k. Whenever (i) or (ii) occurs, we decrease the index j of the function currently processed; thus, (i) and (ii) happen at most n times. If there are only n push operations, there can be no more than n pop operations either, because each pop is sucessful. Thus, the loop is carried out at most $O(n)$ times, which proves the linear time bound. Upon termination, stack S contains the segments of the upper envelope, with the leftmost segment on top. □

This completes the proof of Theorem 1. Finally, we construct the upper envelope in linear time and find its maximum among all integer arguments in $[1, n]$, as explained at the end of Section 2.

Applied to strict lines there is an interesting relation to a variant of *Graham's scan* algorithm for computing the convex hull of a set of sorted points. If *Andrew's monotone chain method* [1] is translated into the dual space, we obtain the above algorithm for a set of strict lines sorted by slope.

4 Extensions

The most simple extension of a one-way path might be a star-shaped directed network as depicted in Figure 4. For a center vertex v we have s incoming path of complexity (number of nodes) m_i for $i = 1, \ldots, s$ and r outgoing paths of complexity n_j for $j = 1, \ldots, r$, as depicted in Figure 4. There are three possible locations for a shortut. The shortcut might be placed fully inside an incoming path, fully inside an outgoing path or it might connect an incoming path with an outgoing path. For the first two cases we can ap-

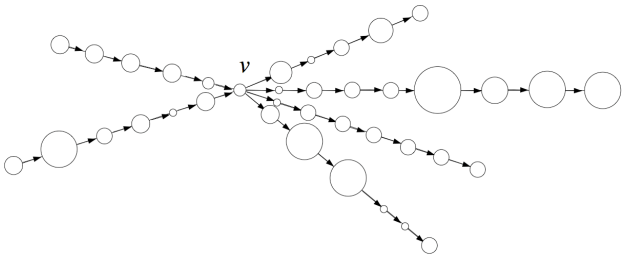


Figure 4: A directed star network with center v, 2 incoming and 4 outgoing paths of different length.

ply the same idea as presented above. We only have to sum up all weights of the incoming paths and all weights of the outgoing paths at v in order to adapt the functions $f_l(k)$ properly. Then we let our algorithm run on every path and obtain a total running time of $\sum_{i=1}^{s} m_i + \sum_{j=1}^{r} n_j$ which is optimal.

So the remaining task is to compute the best shortcut connecting an incoming with an outgoing path. In a direct way we could apply our algorithm to all $s \times r$ alternatives which results in running time $\sum_{i,j}(m_i + n_j)$, obviously this is quadratic. We can improve on this.

First, let us assume that there is only a single incoming path with complexity m. Applying the algorithm for all r outgoing paths using the same incoming path r times gives overall running time $\sum_{j=1}^{r} n_j + r \cdot m$. For a set of s incoming paths we intuitively will collect all incoming paths in a single incoming path of length $\max m_i$. Then the running time for computing the best shortcut between an incoming and an outgoing path is given by $\sum_{j=1}^{r} n_j + r \cdot \max m_i$.

The idea depends on the special nature of the functions $f_l(k)$. Within the algorithm of Section 3 we have to compute intersections in order to compute the upper envelope. In our special case we rather have computed intersection indices. Computing an intersection index means that we have to find out where $f_{l_1}(k) - f_{l_2}(k)$ changes its sign for some $1 \le k < l_1 < l_2 \le n$. From Equation (2) we obtain that $f_{l_1}(k) - f_{l_2}(k)$ for some index k is given by $A_k \cdot ((l_2 - 1 + k)B_{l_2} - (l_1 - 1 + k)B_{l_1})$. Fortunately, the sign of this expression is independent from A_k since

A_k is always positive. Furthermore $(l_2 - 1 + k)$ and $(l_2 - 1 + k)$ are only path distances from v_{l_1} and v_{l_2} to vertex v_k, respectively. This means that knowing the exact value of the indices l_1 and l_2 and k with respect to n is not necessary for computing intersections. Therefore we can apply the above algorithm for an outgoing path (starting from the end and ending at v) without knowing the exact path length n of the path in the beginning and without knowing the exact values A_k to the left. Intersection indices are computed as path distances relative to the given position in the path. In principle we rather compute functions $f_{v_l}(k)$ instead of functions $f_l(k)$ where k is the path distance of a directed path from some vertex w to v_l. The intersection indices computed for a single outgoing path are the same for all incoming paths.

Of course, at the end we have to compute the envelopes to obtain the maximum. At this point we have to make use of the values A_k. A *collected* incoming path will simply represent the maximum values A_k for a given path distance from v. The construction of the single incoming path works as follows. Since we have to compare all incoming paths, we have to be independent from indices. Let v_k be a vertex of an incoming path. Instead of A_k we will denote the sum of weights *arriving* at v_k by A_{v_k}. For any path distance d we compute the vertex $w(d)$ of an incoming path with path distance d to v that has maximum weight $A_{w(d)}$ among all such vertices of the same path distance d from v on all incoming paths. It is easy to see that we can compute these vertices in overall running time $\sum_{i=1}^{s} m_i$. Now the *collected* incoming path is given by vertices $w(\max m_i), w(\max m_i - 1), \ldots, w(1)$ in this order with corresponding weight sums $A_{w(k)}$.

Theorem 4 *Given a star-shaped directed network with s incoming paths of complexity n_i for $i = 1, \ldots, s$ and r outgoing paths of complexity m_j for $j = 1, \ldots, r$. The optimal shortcut can be computed in $\sum_{i=1}^{s} m_i + \sum_{j=1}^{r} n_j + r \cdot \max m_i$ time.*

Finally, we consider the undirected case of a single path. As already depicted in Figure 1 there are different parts of the path that might profit from the shortcut between v and v' or v_k and v_l. A corresponding function $\phi(k, l)$ for indices $l < k$ becomes more complicated. Two vertices profit, if the path length along the shortcut is smaller than the original path length. We collect the benefit of a shortcut in the following functions due to the vertices that benefit.

First formula (from left to right as before):

$$\left(\sum_{i=1}^{k} w_i\right) \times \left(\sum_{j=l}^{n} w_j\right) \times (l - k - 1) \qquad (4)$$

Second formula (from left to inner right):

$$\left(\sum_{i=1}^{k} w_i\right) \times \left(\sum_{j=\lceil \frac{l+k}{2} + 1 \rceil}^{l-1} w_j \times (2j - l - k - 1)\right) \qquad (5)$$

Third formula (from right to inner left):

$$\left(\sum_{i=l}^{n} w_i\right) \times \left(\sum_{j=k+1}^{\lfloor \frac{l+k}{2} - 1 \rfloor} w_j \times (l + k - 2j - 1)\right) \qquad (6)$$

Fourth formula (from inner right to inner left):

$$\sum_{i=\lceil \frac{l+k}{2} + 1 \rceil + 1}^{l-1} w_i \;\times$$
$$\left(\sum_{j=k+1}^{i - \lceil \frac{l+k}{2} + 1 \rceil + k} w_j \times (k - l + 2i - 2j - 1)\right) \qquad (7)$$

Now let $\phi(k, l)$ be the sum of the function (4),(5),(6) and (7). It can be shown that one can choose weights so that two functions $\phi(k, l_1)$ and $\phi(k, l_2)$ will have more than one intersection. This means that the key idea of Theorem 1 does not apply directly to undirected paths. Note, that the best shortcut can be easily computed in $O(n^2)$ time anyway.

5 Conclusion

We have shown how to find an optimum shortcut in an oriented path with weighted vertices, by efficiently constructing upper envelopes of pseudo-lines. In case of a star-shaped directed network the algorithm can be applied also. Natural questions are if one can find efficient algorithms for more general graph classes; even the case of undirected paths is open.

Acknowledgements: We would like to thank an anonymous referee for pointing out some interrelation to a variant of *Graham's scan*; see Section 3.

References

[1] A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. Inform. Process. Lett. 9(5), pp 216–219, 1979

[2] M. Farshi, P. Giannopoulos, and J. Gudmundsson. Improving the stretch factor of a geometric graph by edge augmentation. SIAM Journal on Computing, 38(1), pp. 226–240, 2008.

[3] M. Fischetti, G. Lancia, and P. Serafini. Exact Algorithms for Minimum Routing Cost Trees. Networks 39(3), pp. 161–173, 2002.

[4] J. E. Goodman. Pseudoline Arrangements. In J. E. Goodman and J. ORourke (eds.) Handbook of Comb. and Computat. Geometry, CRC Press, 1997.

[5] M. Sharir and P. Agarwal. Davenport-Schinzel Sequences and Their Geometric Applications. Cambridge University Press, 1995.

[6] B. Ye Wu and K.-M. Chao. Spanning trees and optimization problems. Chapman & Hall/CRC, 2004.

[7] Wu, Bang Ye and Lancia. A Polynomial-Time Approximation Scheme for Minimum Routing Cost Spanning Trees. SIAM J. Comput. 29(3), pp. 761–778, 2000.

Probabilistic matching of solids in arbitrary dimension

Daria Schymura*

Abstract

We give simple probabilistic algorithms that approximately maximize the volume of overlap of two solid shapes under translations and rigid motions. The shapes are subsets of $\mathbb{R}^d$, $d \geq 2$. The algorithms approximate w.r.t. an prespecified additive error and succeed with high probability. Apart from measurability assumptions, we only require from the shapes that uniformly distributed random points can be generated. An important example are finite unions of simplices that have pairwise disjoint interiors.

1 Introduction

We design and analyze simple probabilistic algorithms for matching solid shapes in $\mathbb{R}^d$ under translations and rigid motions. An important example of solid shapes are finite unions of simplices that have pairwise disjoint interiors. On the input of two shapes A and B, the algorithms compute a transformation t^* such that the volume of overlap of $t^*(A)$ and B is approximately maximal.

First, we explain the algorithm and its analysis for the case of translations. In Section 4, we show how to generalize the results to rigid motions.

For translations, the idea of the algorithm is as follows. Given two shapes A and B, a point $a \in A$ and a point $b \in B$ are picked uniformly at random. This tells us that the translation t that is given by the vector $b - a$ maps some part of A onto some part of B. We record this as a "vote" for t and repeat this procedure very often. Then we determine the densest cluster of the resulting point cloud of translation vectors, and output the center of this cluster. This translation maps a large part of A onto B. The details of the algorithm are explained in Section 2.

We show that this algorithm approximates the maximal volume of overlap under translations. More precisely, let t^{opt} be a translation that maximizes the volume of overlap of A and B, and let t^* be a translation that is computed by the algorithm. Given an error bound ε and an allowable probability of failure p, both between 0 and 1, we show bounds on the required number of random experiments, guaranteeing that the difference between approximation and optimum $|t^{\mathrm{opt}}(A) \cap B| - |t^*(A) \cap B| \leq \varepsilon |A|$ with probability at least $1 - p$. Here $|\cdot|$ denotes the volume (Lebesgue measure) of a set. We use $\varepsilon|A|$ and not just ε as error bound because the inequality should be invariant under scaling of both shapes with the same factor.

In a previous publication [1] we considered the 2-dimensional case. Here we not only generalize the results to higher dimensions, but we also give new proofs that improve the bounds on the number of random samples.

Furthermore we considerably improve the time complexity of the algorithm by showing that a simpler definition of a cluster suffices to guarantee approximation. In [1], a translation whose neighborhood contained the maximal number of "votes" was computed, which boiled down to computing a deepest cell of an arrangement of boxes. For N boxes, the best known time bound is $O(N^{d/2}/(\log N)^{d/2-1} \operatorname{polyloglog} N)$ [2]. Here we show that it is sufficient to output the "vote" whose neighborhood contains the maximal number of "votes", which can be computed by brute force in time $O(N^2)$ in any dimension. The time bound can be further improved to $O(N(\log N)^{d-1})$ by using orthogonal range counting queries [4].

Cheong et al. [3] introduce a general probabilistic framework, which they use for approximating the maximal area of overlap of two unions of n and m triangles in the plane, with prespecified absolute error ε, in time $O(m+(n^2/\varepsilon^4)(\log n)^2)$ for translations and in time $O(m + (n^3/\varepsilon^8)(\log n)^5)$ for rigid motions. The latter time bound is smaller in their paper, due to a calculation error in the final derivation of the time bound, as was noted in [11]. Their algorithm works with high probability.

For two simple polygons with n and m vertices in the plane, Mount et al. [8] show that a translation that maximizes the area of overlap can be computed in time $O(n^2m^2)$.

For maximizing the volume of overlap of two unions of simplices under rigid motions, no exact algorithm that runs in polynomial time is known, not even in the plane. Vigneron gives an FPTAS with relative error ε for dimension $d \geq 2$ [11]. For two polyhedra P and Q in $\mathbb{R}^d$, given as the union of m and n simplices, respectively, the algorithm for approximating the maximal volume of overlap has time complexity $O\big((\frac{nm}{\varepsilon})^{d^2/2+d/2+1}(\log \frac{nm}{\varepsilon})^{d^2/2+d/2+1}\big)$, which can be improved to $O\big((n^6/\varepsilon^3)\log^4(n/\varepsilon)\beta(n/\varepsilon)\big)$ in the plane where β is a very slowly growing function related to the inverse Ackermann function.

*Institute of Computer Science, Freie Universität Berlin, schymura@inf.fu-berlin.de

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

2 A probabilistic algorithm for matching under translations

Before stating the main result for translations, we introduce some definitions. The boundary ∂A of a set $A \subseteq \mathbb{R}^d$ is the set of points that are contained in its closure, but not in its interior. We measure the boundary of d-dimensional sets by the $(d-1)$-dimensional Hausdorff measure, and denote it slightly sloppy by $|\partial A|$. For a definition of the Hausdorff measure and related definitions, we refer the reader to [7]. The *isoperimetric quotient* of $A \subset \mathbb{R}^d$ is defined to be ${}^{|\partial A|^d}/_{|A|^{d-1}}$. The isoperimetric quotient can be considered as a certain measure of the *fatness* of a figure A.

We always assume shapes to be Lebesgue measurable subsets of $\mathbb{R}^d$ that have positive, finite volume and whose boundary is measurable by the $(d-1)$-dimensional Hausdorff measure and has positive, finite $(d-1)$-dimensional volume.

Our algorithm can be applied to all shapes from which we can generate random sample points. For unions of simplices, the runtime of our method depends only linearly on the number of vertices, but it depends more significantly on fatness parameters as the isoperimetric quotient of one of the shapes and, in the case of rigid motions, on the ratio ${}^{\operatorname{diam}(A)^d}/_{|A|}$.

Theorem 1 (Runtime for translations) *Let A and B be shapes in constant dimension d, and let $\varepsilon, p \in (0,1)$ be parameters. Assume that we are given a lower bound on $|A|$ and an upper bound on $|B|$, which differ only by a constant, and an upper bound K_A on the isoperimetric quotient of A. Assume further that N uniformly distributed random points can be generated from a shape in time $T(N)$.*

Then a translation that maximizes the volume of overlap of A and B up to an additive error of $\varepsilon|A|$ with probability $\geq 1-p$ can be computed in time $O\big(T(N) + N(\log N)^{d-1}\big)$ where $N = O\big(\varepsilon^{-(2d+2)}\, K_A^2\, \log \frac{2}{p}\big)$.

If A and B are finite unions of at most n simplices that have pairwise disjoint interiors, then $T(N) = O(n + N \log n)$.

To prove this theorem, we describe an algorithm, Algorithm 1, and then prove that it is correct and has the runtime claimed in the theorem.

The following theorem gives bounds on the output of ClusteringSize(ε, A) and SampleSize$(B, \varepsilon, \delta, p)$ in Algorithm 1 that guarantee that the output of Algorithm 1 approximates the maximal volume of overlap of A and B up to an additive error of $\varepsilon|A|$ with probability at least $1-p$.

Of course, the roles of A and B can be swapped, which may result in better bounds. For the shortness of presentation, we do not reflect this fact in the following.

Algorithm 1: ProbMatchT

Input: shapes $A, B \subset \mathbb{R}^d$, error bound $\varepsilon \in (0,1)$, allowed probability of failure $p \in (0,1)$
real $\delta \leftarrow$ ClusteringSize(ε, A);
integer $N \leftarrow$ SampleSize$(B, \varepsilon, \delta, p)$;
collection $Q \leftarrow \emptyset$;
for $i = 1 \dots N$ **do**
  point $a \leftarrow$ randomPoint(A);
  point $b \leftarrow$ randomPoint(B);
  add$(Q, b - a)$;
end
return *FindDensestClusterT*(Q, δ);

Function `FindDensestClusterT`(Q, δ)

Input: collection Q of points in $\mathbb{R}^d$, positive number δ
Output: point t in Q such that the cube of side length 2δ that is centered at t contains a maximal number of points from Q

Theorem 2 (Correctness of Algorithm 1) *Let A and B be shapes in constant dimension d, and let $\varepsilon, p \in (0,1)$ be parameters. If ClusteringSize(ε, A) returns a positive number $\delta \leq \frac{d}{d-1} \cdot \frac{\sqrt{2}}{3\sqrt{d}} \cdot \varepsilon \cdot \frac{|A|}{|\partial A|}$ and SampleSize$(B, \varepsilon, \delta, p)$ returns an integer $N \geq C\varepsilon^{-2}\, \delta^{-2d}\, |B|^2\, \log \frac{2}{p}$ for some universal constant $C > 0$, then Algorithm 1 computes on the input (A, B, ε, p) a translation that maximizes the volume of overlap of A and B up to an additive error of $\varepsilon|A|$ with probability at least $1-p$.*

The universal constant C can be deduced from the proofs. Note that, when both shapes are scaled with the same factor, the sample size N does not change. In other words, it is homogeneous. The clustering size scales with the shapes. This is reasonable because blowing up the shapes coarsens fine features.

3 Proof idea of Theorem 2

Let μ be the probability distribution on the translation space that is induced by the random experiment in the algorithm, and let μ_N be the empirical measure. Recall that μ has a *density function* f if the probability $\mu(E)$ of any event $E \subseteq \Omega$ can be computed as the integral over the density function $\int_E f(x)dx$.

The main idea is to prove that the density function of μ is proportional to the objective function, that is the function that maps a translation vector $t \in \mathbb{R}^d$ to the volume of the intersection of $t(A)$ and B. Thus the goal is to find a translation at which the density function is approximately maximal.

Conceptually, the density function is approximated in a two step process. Let $B(t, \delta)$ be a ball of ra-

dius δ, centered at t, w.r.t. the metric that is induced by the maximum norm. First, $f(t)\cdot|B(t,\delta)|$ is close to $\mu(B(t,\delta))$ if f is nice enough and δ is sufficiently small. Second, the probability of a small cube $\mu(B(t,\delta))$ is close to $\mu_N(B(t,\delta))$ if N is sufficiently large.

The analysis of the algorithm is based on these simple ideas whose details are not that easy. They are hidden in the following longish theorem that follows from theorems in Chapters 3 and 4 of [5].

Theorem 3 (Probabilistic toolbox) *Let $\Omega \subseteq \mathbb{R}^k$ be a metric space, and let $\mathcal{B}$ be the set of balls of some fixed radius $\delta > 0$ in Ω. Let* vol *be a measure on Ω such that, for all $x \in \Omega$, the volume* $\mathrm{vol}(B(x,\delta)) = v_\delta$ *for some $v_\delta > 0$. Assume further that $\mathcal{B}$ has finite VC dimension V.*

Let μ be a probability measure on Ω that has a Lipschitz continuous density function f with Lipschitz constant L. Let $X_1, \ldots, X_N$ be i.i.d. random variables taking values in Ω with common distribution μ and empirical measure μ_N.

Let $j \in \{1, \ldots, N\}$ be such that $\mu_N(B(X_j,\delta)) = \max_{1\le i\le N} \mu_N(B(X_i,\delta))$. Then, for all $\tau > 0$ and for all $x \in \Omega$, with probability $\ge 1 - 2e^{-2N\tau^2}$, we have $f(X_j) \ge f(x) - 2(c\sqrt{V/N} + \tau)/v_\delta - 3L\delta$.

In this inequality, c is a universal constant.

The key lemma to apply Theorem 3 states that the density function of μ is proportional to the objective function. It follows from a transformation rule for density functions.

Lemma 4 (Key lemma) *Let X be the random vector on $\mathbb{R}^d$ that draws translations $t = b - a$ where $(a,b) \in A \times B \subset \mathbb{R}^{2d}$ is drawn uniformly at random. The density function of X is given by $f(t) = \frac{|t(A)\cap B|}{|A|\cdot|B|}$.*

Furthermore we have to show that the density function f is Lipschitz continuous. In the following theorem, let $\mathcal{H}^k$ denote the k-dimensional Hausdorff measure. For Lebesgue measurable sets in $\mathbb{R}^d$, the d-dimensional Hausdorff measure and the Lebesgue measure coincide. The symmetric difference is denoted by $\triangle$.

Theorem 5 *[9] Let $A \subset \mathbb{R}^d$ be bounded. Let $t \in \mathbb{R}^d$ be a translation vector.*

Then $\mathcal{H}^d(A \triangle (A+t)) \le |t|\,\mathcal{H}^{d-1}(\partial A)$.

This implies that the density function f is Lipschitz continuous. Applying the Cauchy-Schwarz inequality yields $||t-s||_2 \le \sqrt{d}||t-s||_\infty$, which is best possible.

Corollary 6 *The function f on $\mathbb{R}^d$ that is given by $f(t) = \frac{|t(A)\cap B|}{|A|\cdot|B|}$ is Lipschitz continuous with constant $L = \frac{\sqrt{d}\,|\partial A|}{2|A|\cdot|B|}$ w.r.t. the metric that is induced by the maximum norm.*

With these results, the proof of Theorem 2 is an application of Theorem 3. Note that the VC dimension of the class of rectangles in $\mathbb{R}^d$ equals $2d$ [5].

4 Rigid motions

A rigid motion r on $\mathbb{R}^d$ is given by $r(x) = Mx + t$ where $M \in \mathbb{R}^{d\times d}$ is a rotation matrix and $t \in \mathbb{R}^d$ is a translation vector. A matrix M is contained in the group of rotation matrices $SO(d) \subset \mathbb{R}^{d\times d}$ if it is orthogonal, meaning $M^T = M^{-1}$, and $\det M = 1$. We identify each rigid motion with the pair of its rotation matrix and translation vector. Thus the space of rigid motions equals $SO(d) \times \mathbb{R}^d$.

The algorithm for rigid motions works similarly as the algorithm for translations. We draw a rotation matrix $M \in SO(d)$, a point $a \in A$ and a point $b \in B$ uniformly at random. Then we register the unique rigid motion that has M as rotation matrix and maps a onto b as a "vote" in the transformation space. After many rounds, say N, we determine the best cluster in the space of rigid motions.

There are many different methods to compute a random rotation matrix described in the literature; see for example [6, 10]. To define the uniform distribution formally, a volume has to be defined. The group $SO(d)$ is $\binom{d}{2}$-dimensional. The volume $|\cdot|$ in $SO(d)$ is measured by the $\binom{d}{2}$-dimensional Haar measure.

For a matrix $M = (m_{ij})_{1\le i,j\le d}$, let $||M||_2 = \sqrt{\sum_{1\le i,j\le d}(m_{ij})^2}$ be the Frobenius norm. Denote the Euclidean norm on $\mathbb{R}^d$ also by $||\cdot||_2$. Define $B_2(M,\delta)$ and $B_2(t,\delta)$ to be the closed balls of radius δ w.r.t. the metrics induced by the Frobenius and the Euclidean norm.

We define a *δ-neighborhood* of a rigid motion (M,t) by $B((M,t),\delta) = B_2(M, \delta/\operatorname{diam}(A)) \times B_2(t,\delta)$. The radius of the rotational part of the neighborhood depends on the diameter of A because it should not change if A is scaled. The "rotational distance" of shapes does not depend on their absolute size. A best cluster is the random rigid motion whose neighborhood contains the maximal number of other random rigid motions. We give a pseudocode description of the algorithm for rigid motions, Algorithm 2.

Theorem 7 (Correctness of Algorithm 2) *Let A and B be shapes in constant dimension d, and let $\varepsilon, p \in (0,1)$ be parameters.*

There are constants $C, C' > 0$ such that, if ClusteringSize(ε, A) returns a positive $\delta \le C\varepsilon\frac{|A|}{|\partial A|}$ and SampleSize($B, \varepsilon, \delta, p$) returns an integer $N \ge C'\varepsilon^{-2}\,\delta^{-d^2-d}\,\operatorname{diam}(A)^{-d^2+d}\,|B|^2\,\log\frac{2}{p}$, then Algorithm 2 computes on the input (A, B, ε, p) a rigid motion that maximizes the volume of overlap of A and B up to an additive error of $\varepsilon|A|$ with probability $\ge 1-p$.

Algorithm 2: ProbMatchRM

Input: shapes $A, B \subset \mathbb{R}^d$, error bound $\varepsilon \in (0,1)$, allowed probability of failure $p \in (0,1)$
real $\delta \leftarrow$ ClusteringSize(ε, A);
integer $N \leftarrow$ SampleSize$(B, \varepsilon, \delta, p)$;
collection $Q \leftarrow \emptyset$;
for $i = 1 \dots N$ **do**
  rotation matrix $M \leftarrow$ randomRotation();
  point $a \leftarrow$ randomPoint(A);
  point $b \leftarrow$ randomPoint(B);
  add$(Q, (M, b - Ma))$;
end
return *FindDensestClusterRM*$(Q, \delta, \mathrm{diam}(A))$;

Function `FindDensestClusterRM`(Q, δ, Δ)

Input: collection Q of points in $\mathbb{R}^{d\times d} \times \mathbb{R}^d$, positive numbers δ and Δ
Output: point (M, t) in Q such that the neighborhood $B_2(M, \delta/\Delta) \times B_2(t, \delta)$ of (M, t) contains a maximal number of points from Q

Theorem 8 (Runtime for rigid motions) *Let A, B, ε, and p be given under the same assumptions as in Theorem 1. Additionally, let D_A be given such that $\frac{\mathrm{diam}(A)^d}{|A|} \le D_A$. Then a rigid motion that maximizes the volume of overlap of A and B up to an additive error of $\varepsilon|A|$ with probability $\ge 1 - p$ can be computed in time $O\big(T(N) + N^2\big)$ where $N = O\big(\varepsilon^{-(d^2+d+2)}\, K_A^{d+1}\, D_A^{d-1}\, \log \frac{2}{p}\big)$.*

As in the case of translations, the density function on the transformation space is proportional to the objective function, and it is Lipschitz continuous:

Lemma 9 *Let Z be the random vector that draws rigid motions $(M, b - Ma) \in SO(d) \times \mathbb{R}^d$ where $(M, a, b) \in SO(d) \times A \times B$ is drawn u.a.r. The density function of Z is given by $g(r) = \frac{|r(A)\cap B|}{|SO(d)|\cdot|A|\cdot|B|}$.*

The function g is Lipschitz continuous with constant $L = \frac{|\partial A|}{|SO(d)|\cdot|A|\cdot|B|}$ w.r.t. the metric $d(r,s) = \max\{\, \mathrm{diam}(A) \cdot ||M - N||_2\,,\, ||p - q||_2 \,\}$ for rigid motions $r = (M, p)$ and $s = (N, q)$.

Observe that a δ-neighborhood of a rigid motion, as defined above, equals the closed ball of radius δ w.r.t. the metric d. In order to apply Theorem 3, one has to prove that all neighborhoods have the same volume, which follows from the fact that the $\binom{d}{2}$-dimensional Hausdorff measure is a Haar measure on $SO(d)$. Additionally, one has to compute a lower bound on the volume of such a neighborhood.

The Lipschitz continuity of g can be deduced using Theorems 5, 10 and the fact that the volume of the symmetric difference fulfills the triangle inequality. We assume that A and B have $(\mathcal{H}^{d-1}, d-1)$-rectifiable boundaries.

Theorem 10 *[9] Let $A \subset \mathbb{R}^d$ be bounded. Let $M \in \mathbb{R}^{d\times d}$ be a rotation matrix and let $w = \max_{a\in\partial A} ||a - Ma||_2$.*

Then $\mathcal{H}^d(A \triangle MA) \le \left(2d/d+1\right)^{\frac{d-1}{2}} \cdot w \cdot \mathcal{H}^{d-1}(\partial A)$.

The constant $\left(2d/d+1\right)^{\frac{d-1}{2}}$ can be replaced by 1 for sets that have an $(\mathcal{H}^{d-1}, d-1)$-rectifiable boundary.

In the 2-dimensional case, the results can be significantly improved by representing a rigid motion by the pair of its rotation angle (instead of the rotation matrix) and the translation vector, as it was done in [1].

References

[1] H. Alt, L. Scharf, and D. Schymura. Probabilistic matching of planar regions. *Computational Geometry, Theory and Applications (CGTA)*, 43:99–114, 2010. Special Issue on the 24th European Workshop on Computational Geometry (EuroCG'08).

[2] T.M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Computational Geometry*, 43(3):243 – 250, 2010.

[3] O. Cheong, A. Efrat, and S. Har-Peled. Finding a guard that sees most and a shop that sells most. *Discrete and Computational Geometry*, 37:545–563, 2007.

[4] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry*. Springer, third edition, 2008.

[5] L. Devroye and G. Lugosi. *Combinatorial Methods in Density Estimation*. Springer, 2001.

[6] R.M. Heiberger. Generation of random orthogonal matrices. *Applied Statistics*, 27:199–206, 1978.

[7] H. Federer. *Geometric Measure Theory*. Springer, 1969.

[8] D.M. Mount, R. Silverman, and A.Y. Wu. On the area of overlap of translated polygons. *Computer Vision and Image Understanding: CVIU*, 64:53–61, 1996.

[9] D. Schymura. An upper bound on the volume of the symmetric difference of a body and a congruent copy. *ArXiv e-prints*, 2010. `http://arxiv.org/abs/1010.2446`.

[10] M.A. Tanner and R.A. Thisted. A remark on AS127. Generation of random orthogonal matrices. *Applied Statistics*, 31:190–192, 1982.

[11] A. Vigneron. Geometric optimization and sums of algebraic functions. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 906–917, 2010.

Minimum-Link Paths Revisited*

Joseph S. B. Mitchell† Valentin Polishchuk‡ Mikko Sysikaski†

Abstract

A path or a polygonal domain is C-oriented if the orientations of its edges belong to a set of C given orientations; this is a generalization of the notable rectilinear case ($C = 2$). We study exact and approximation algorithms for minimum-link C-oriented paths and paths with unrestricted orientations, both in C-oriented and in general domains. We obtain the following results:

- An optimal algorithm for finding a minimum-link rectilinear path in a rectilinear domain; using only elementary data structures, our algorithm and its analysis are simpler than the earlier ones of [Sato, Sakanaka and Ohtsuki, 1987], [Das and Narasimhan, 1991].
- An algorithm to find a minimum-link C-oriented path in a C-oriented domain; our algorithm is simpler and more time-space efficient than the earlier one of [Adegeest, Overmars and Snoeyink, 1994].
- An extension of our techniques to find a C-oriented minimum-link path in a general (not necessarily C-oriented) domain.
- 3SUM-hardness of finding a minimum-link path with unrestricted orientations (even in a C-oriented domain). This answers a question from the survey of Mitchell [Goodman and O'Rourke, eds., 1997, 2004] and Problem 22 in The Open Problems Project.

We also present approximation methods for link distance, including the following:

- A more time-space efficient algorithm to find a 2-approximate C-oriented minimum-link path.
- A notion of "robust" paths. We show how minimum-link C-oriented paths approximate the robust paths with unrestricted orientations to within an additive error of 1.
- A subquadratic-time algorithm with a non-trivial approximation guarantee for the general (unrestricted-orientation) minimum-link paths in general domains.

All of our algorithms not only find minimum-link paths but also build, within the same time and space bounds, the corresponding link distance maps—exact or approximate. For instance, using our algorithms, one can construct in subquadratic time linear-size approximate (additive or multiplicative) maps for general minimum-link paths in general domains; this is in contrast with the exact link distance maps, which may have quartic complexity [Suri and O'Rourke, 1986].

1 Introduction

Minimum-link problems arise in motion planning domains were turning is expensive, in line simplification, guarding applications, VLSI, wireless communication, and other areas. An instance of the problem is specified by an n-vertex polygonal domain P with h holes, and two points $s, t \in P$; the goal is to find an s-t path with fewest edges (links). In the query version of the problem, the goal is to build a data structure (link distance map) to answer efficiently link distance queries to s.

We give algorithms for computing exact and approximate C-oriented minlink paths and show how they approximate "robust" paths with unrestricted orientations; we also give approximation algorithms for the general minlink problem.

1.1 Previous work

If P is a simple polygon, a minlink path can be found in linear time [7, 9, 21]. The general idea is to do the "staged illumination" of P [8, Sections 26.4, 27.3] that starts from illuminating the visibility polygon of s. At the beginning of any stage the boundary between the lit and the dark parts of P is a set of "windows"; each window w is a chord cutting out a dark part $P_w \subset P$. The crux of the linear-time algorithm is that P_w is "unique" to w: the part of P_w illuminated at the next stage is exactly what is seen from w (even if a point $p \in P_w$ is seen from another window $w' \neq w$, we do not care about shining light from w' into P_w). At any stage, the illumination is guided by DFS in the dual of a triangulation of P; since the dual is a tree, any triangle is lit only once (triangles intersected by windows are first lit only partially, and are fully lit at the next stage).

In polygonal domains with holes, the algorithm of Mitchell, Rote and Woeginger [16] computes a minlink

*Longer version is available online at: `http://hiit.fi/valentin.polishchuk/ml.pdf`

†Stony Brook University, `jsbm@ams.stonybrook.edu`

‡University of Helsinki, `polishch@cs.helsinki.fi`

path in $O(n^2\alpha^2(n)\log n)$ time. It was believed that a faster algorithm is possible (e.g., in [6, p. 263] the result of [16] is called "suboptimal"). Nevertheless, the only previously known lower bound, also due to [16], was $\Omega(n \log n)$; the same bounds for the *rectilinear* case are given in [6, 14]. Also, no subquadratic-time approximation algorithm is known.

Robustness of paths was previously addressed in terms of *distance* tolerance by employing high-clearance [18, 24, 25] or thick-paths [2, 5, 13, 17] models. Similar, distance-based robustness was considered in the context of *curvature-constrained* paths in [3, 12, 23].

Minimum-link C-oriented paths in C-oriented domains were studied by Adegeest, Overmars and Snoyeink [1]. Two algorithms are presented in [1]: one running in $O(C^2 n \log n)$ time and space, the other—in $O(C^2 n \log^2 n)$ time and $O(C^2 n)$ space. Both algorithms build C trapezoidations of the domain and label the trapezoids with link distance from s; this way an $O(Cn)$-space structure is created that answers link distance queries in $O(C \log n)$ time. The labeling of the trapezoids proceeds in n *steps*, with label-k trapezoids receiving their label at step k. By induction on k, any such trapezoid must be intersected by a label-$(k-1)$ trapezoid of a different orientation; hence, step k boils down to detecting all unlabeled trapezoids intersected by label-$(k-1)$ trapezoids. In terms of the *intersection graph* of trapezoids (which has nodes corresponding to trapezoids and edges corresponding to intersecting pairs of trapezoids), the labeling is the BFS starting from the C-oriented segments through s.

The idea of performing an efficient BFS in the intersection graph without building the (potentially quadratic-size) graph explicitly, dates back to the work on minimum-link *rectilinear* paths [6, 10, 11, 14, 15, 19, 20, 26, 27]. Imai and Asano's [10, 11] data structure allows one to do the BFS—and hence find a minlink rectilinear path—in $O(n \log n)$ time and space. Still, it was not until the work of Das and Narasimhan [6] (and the lesser known work of Sato, Sakanaka and Ohtsuki [20]) that an optimal, $O(n \log n)$-time $O(n)$-space BFS implementation was developed. In the implementation, one step of the BFS is reduced to a pair of sweeps—the UpSweep and the DownSweep.

2 Overview of the results

Rectilinear paths in rectilinear domains The data structures employed in [6, 20] are simpler than the (much more general-purpose) structure of Imai–Asano [10, 11], but they are still more complicated than one would hope for the basic problem of finding rectilinear paths amidst rectilinear obstacles. We present a simplified implementation of the BFS step in the intersection graph; our modification does not affect the asymptotic time and space optimality of the algorithm. The crux of the simplification is the use of a *single* tree for storing the intersection of the sweepline with the trapezoids that were lit at the previous step.

Another, minor modification present in our algorithm comes from performing only one, upward sweep, at any step of the BFS. The sweep starts from what we call the "pot" trapezoids—those into which the different-orientation trapezoids lit at the previous step are "planted". The planting is nothing but a means to initialize the sweep (without the planting, it is not clear how to discover efficiently even a single edge in the intersection graph). While our modification saves only a factor of 2 in running time, it serves as the basis of our improvements for the general case of C-oriented paths with $C > 2$.

C-oriented paths in C-oriented domains As noted in [1], the efficient methods developed to perform BFS in the trapezoids intersection graph for the rectilinear version do not extend to C-oriented paths when $C > 2$. We look closely at why this is the case. One reason is that for $C > 2$ some trapezoids may get labeled only partially during a BFS step; this complicates the BFS because the intersection graph changes from step to step, and, in the final link distance map, trapezoids may get split into subtrapezoids. The partial labeling and splitting are due to the possibility that two different-orientation trapezoids do not "straddle" each other; instead they both may be "flush" with an obstacle edge whose orientation is different from the orientations of both trapezoids (this was not the case in the rectilinear version since there were only 2 orientations). However, such flushness can be read off easily from lists of incident trapezoids stored with every edge of P; thus, discovering partially labeled trapezoids becomes the easy part of the algorithm.

After the partially labeled trapezoids are processed, we are left with discovering unlabeled trapezoids "fully straddled" by trapezoids labeled at the previous step. As with the rectilinear case, it is the straddling that leads to a superlinear-size intersection graph and makes a subquadratic algorithm less trivial. It is tempting to reuse here the sweeping techniques developed for the rectilinear version; the stumbling block, though, is choosing the direction of the sweep. Indeed, no matter in which direction the sweep proceeds, the intersection of the sweepline with a trapezoid does not change only at discrete "events": while the sweepline intersects a non-parallel side of the trapezoid, the intersection changes continuously. The good news is that this is the *only* reason for a continuous change of the intersection. This prompts to get rid of the non-parallel sides of the trapezoids by clipping them into parallelograms, with the new sides parallel to the

sweepline; after the clipping (and planting the parallelograms appropriately) is done, we are able to reuse the rectilinear-case machinery and finish the BFS step with $C(C-1)$ sweeps—one per pair of orientations. Overall we obtain an $O(C^2 n \log n)$-time $O(Cn)$-space algorithm.

C-oriented paths in general domains We generalize our techniques to compute the C-oriented link distance map also in domains with *unrestricted* orientations of edges. We observe that the algorithm for C-oriented path in a C-oriented domain uses C-orientedness of the domain only to bound the complexity of the final map (without the C-orientedness, the complexity may blow up). However, the blowup happens only "deep inside" of certain trapezoids. Thus, we declare the deep parts of such trapezoids as a separate cell. The path to a query point q inside the cell consists of 2 parts: path from s to enter the trapezoid, and a "zigzag" of extreme orientations to q. The number of links in the first part is given by the usual map, the number of links in the second part can be determined in constant time (assuming constant-time floor function) because of the regular pattern of the path—it bounces off the sides of the trapezoid until reaching the query point.

Unrestricted paths Our focus on C-oriented path is partially justified by observing that the general minlink path problem is 3SUM-hard—even in C-oriented domains.

2.1 Approximations

2-approximate C-oriented paths The C-oriented link distance may be approximated by requiring that every second link of the path is horizontal. To find a minlink C-oriented path with this requirement one can do the BFS in the trapezoids intersection graph with one modification: instead of checking for intersection between trapezoids for all pairs of orientations, only check for intersections between the horizontal trapezoids and the other ones. Such a modification decreases the running time of our algorithm to $O(Cn \log n)$, but the space remains $O(Cn)$ because the C trapezoidations of P are still constructed.

To reduce the space to $O(n)$ we only do the horizontal trapezoidation, and go back to the rectilinear-case ideas of Das and Narasimhan [6] who label the horizontal trapezoids *without* using the vertical ones (in our case, we label the horizontal trapezoids without using *any* other ones). In particular, without the other trapezoidations we cannot use our planting (there is simply nothing to plant!) to initialize the sweep; thus we do both the UpSweep and the DownSweep à la [6] starting from trapezoids labeled on the previous step.

Approximating robust paths with C-oriented paths We define *robust* paths as those whose edges may be "wiggled" without intersecting the obstacles. We prove that C-oriented paths can approximate a robust minlink path to within a one-sided additive error of 1: we show how to build a data structure to answer efficiently the approximate robust link distance queries; to output the approximating C-oriented path itself takes additional time proportional to the link distance. We emphasize that we compute the data structure for approximate link distance queries in *subquadratic time*, which compares favorably with the $\Theta(n^4)$ worst-case complexity of the *exact* map [22].

$O(\sqrt{h})$-approximation for unrestricted paths We show that in $O(n\sqrt{h} + n \log n + h^{3/2} \log h)$ time one can find an $O(\sqrt{h})$-approximate minlink path with arbitrary orientations; this is the first subquadratic-time approximation algorithm with a non-trivial approximation guarantee for the general case. In fact, our algorithm builds a linear-size $O(\sqrt{h})$-approximate link distance map (which again compares well to the $\Theta(n^4)$-size exact map).

The two ingredients of our approximation are low-stabbing-number bridges and staged illumination (window partition) in simple polygons. We bridge the holes to the outer boundary of P, thereby obtaining a simple polygon. We do not make the bridges fully opaque; rather, they are "semi-transparent": we triangulate the simple polygon and do the staged illumination from s, but as we go, each time a bridge is illuminated on one of its sides, at the next step we consider it to be illuminated also on its other side, and continue the staged illumination. In comparison to opt, we are delayed by 1 link every time an edge of opt crosses a bridge; i.e., on every edge of opt we have as many additional vertices as there are bridges that the edge crosses. Using a low-stabbing-number tree for the bridging [4], we ensure that each edge of opt crosses $O(\sqrt{h})$ bridges, and thus we obtain an $O(\sqrt{h})$ multiplicative approximation.

As described above, the illumination takes $O(nh)$ time, since a triangle can be discovered by light emanating from h bridges. To address this inefficiency, we declare a triangle opaque as soon as it is lit $m \leq h$ times. We prove that with the right choice of m this does not delay the illumination by too much, while decreasing the runtime of the illumination to $O(nm)$.

References

[1] J. Adegeest, M. H. Overmars, and J. Snoeyink. Minimum-link c-oriented paths: Single-source queries. *Int. J. Comput. Geometry Appl.*, 4(1):39–51, 1994.

[2] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk.

Maximum thick paths in static and dynamic environments. *Computational Geometry Theory and Applications*, 43(3):279–294, 2010.

[3] J. Barraquand and J.-C. Latombe. Nonholonomic multi-body mobile robots: controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[4] B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.

[5] L. P. Chew. Planning the shortest path for a disc in $O(n^2 \log n)$ time. In *Proceedings of the 1st Annual Symposium on Computational Geometry*, pages 214–220, 1985.

[6] G. Das and G. Narasimhan. Geometric searching and link distance. *Algorithms and Data Structures*, pages 261–272, 1991.

[7] S. K. Ghosh. Computing the visibility polygon from a convex set and related problems. *J. Algorithms*, 12(1):75–95, 1991.

[8] J. E. Goodman and J. O'Rourke. *Handbook of discrete and computational geometry*. CRC Press series on discrete mathematics and its applications. Chapman & Hall/CRC, 2004.

[9] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom.*, 4:63–97, 1994.

[10] H. Imai and T. Asano. Efficient algorithms for geometric graph search problems. *SIAM J. Comput.*, 15(2):478–494, 1986.

[11] H. Imai and T. Asano. Dynamic orthogonal segment intersection search. *J. Algorithms*, 8(1):1–18, 1987.

[12] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In Z. Li and J. F. Canny, editors, *Nonholonomic Motion Planning*, pages 271–342. Kluwer Academic Pubishers, Norwell, MA, 1992.

[13] I. Kostitsyna and V. Polishchuk. Simple wriggling is hard unless you are a fat hippo. In *Fifth International Conference on Fun with Algorithms (FUN)*, 2010.

[14] D. T. Lee, C. D. Yang, and C. K. Wong. Rectilinear paths among rectilinear obstacles. *Discrete Appl. Math.*, 70:185–215, 1996.

[15] A. Maheshwari, J.-R. Sack, and D. Djidjev. Link distance problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry, Chapter 12*, pages 519–558. Elsevier Science, Amsterdam, 2000.

[16] J. Mitchell, G. Rote, and G. Woeginger. Minimum-link paths among obstacles in the plane. *Algorithmica*, 8(1):431–459, 1992.

[17] J. S. B. Mitchell and V. Polishchuk. Thick noncrossing paths and minimum-cost flows in polygonal domains. In J. Erickson, editor, *Symposium on Computational Geometry*, pages 56–65. ACM, 2007.

[18] C. Ó'Dúnlaing and C.-K. Yap. A "retraction" method for planning the motion of a disc. *J. Algorithms*, 6(1):104–111, 1985.

[19] T. Ohtsuki. Gridless routers - new wire routing algorithm based on computational geometry. In *Internat. Conf. on Circuits and Systems, China*, 1985.

[20] M. Sato, J. Sakanaka, and T. Ohtsuki. A fast line-search method based on a tile plane. In *Proc. IEE ISCAS*, pages 588–591, 1987.

[21] S. Suri. A linear time algorithm with minimum link paths inside a simple polygon. *Comput. Vision Graph. Image Process.*, 35(1):99–110, 1986.

[22] S. Suri and J. O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *Proceedings of the second annual symposium on Computational geometry*, SCG '86, pages 14–23, New York, NY, USA, 1986. ACM.

[23] H. Wang and P. K. Agarwal. Approximation algorithms for curvature constrained shortest paths. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 409–418, 1996.

[24] R. Wein, J. P. van den Berg, and D. Halperin. The visibility-voronoi complex and its applications. *Comput. Geom.*, 36(1):66–87, 2007.

[25] R. Wein, J. P. van den Berg, and D. Halperin. Planning high-quality paths and corridors amidst obstacles. *I. J. Robotic Res.*, 27(11-12):1213–1231, 2008.

[26] C. D. Yang, D. T. Lee, and C. K. Wong. On bends and lengths of rectilinear paths: a graph theoretic approach. *Internat. J. Comput. Geom. Appl.*, 2(1):61–74, 1992.

[27] C. D. Yang, D. T. Lee, and C. K. Wong. Rectilinear paths problems among rectilinear obstacles revisited. *SIAM J. Comput.*, 24:457–472, 1995.

Reconstruction of Flows of Mass under the L1 Metric

Jörg Bernhardt* Stefan Funke† Sabine Storandt†

Abstract

We consider the problem of 'moving mass' under the L1 metric. In contrast to the well-known Earth-Mover's-Distance (EMD, [5]) we are not only interested in the amount of work (aka EM distance) that is necessary to transfer the mass but also in reconstructing (local) translations that lead to this movement of mass. This problem is motivated by an image matching application for analysis of electrophorese gel experiments in biology.

1 Introduction

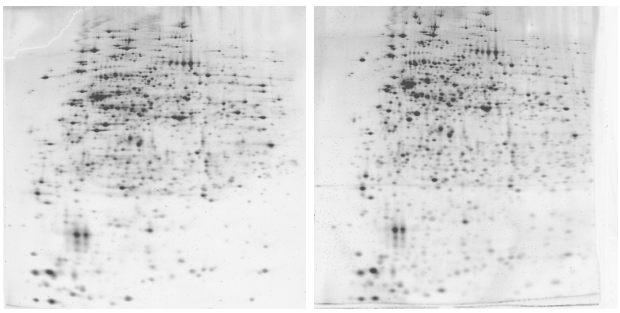


Figure 1: Gel scans for Bacillus Subtilis where we are interested in corresponding spots in both scans.

In the last decade *proteomics* – a branch of molecular biology – has been established as an important area of research. The goal is to detect the totality of proteines – the proteome – inside a cell and identify and quantify single proteins in order to relate them to their functions. For that purpose comparing the proteome of different samples is the method of choice. For example bacteria that suffer from starvation are expected to produce less proteines which are not essential (e.g. for cell division), but instead more proteines that work as enzymes for alternative food sources. In order to detect which protein belongs to which group we have to compare the sample of the starving bacteria to the one of a control group. To that end electrophoresis is a common and cheap method to separate the up to several thousands of proteins that might occur in a cell. Here, in the first dimension isoelectric focusing is used, essentially ordering the proteins according to their pH values. In the second dimension the proteins are then separated according to their masses. The result is a two-dimensional separation of the proteins visualized in a *gel scan*, where proteins can be seen as dark accumulations – so called spots (depicted in Figure 1). The size and the color depth of a spot is proportional to the produced amount of protein. The main difficulty is to identify corresponding *protein spots* in both gel scans. Ideally, if the separation in both dimensions was perfect, each protein would end up at the same position in both scans. Unfortunately, the separation steps are physical processes which are easily disturbed resulting in global *and* local distortions. Hence the identification of those correspondences is highly non-trivial.
To make this problem accessible for mathematical methods, we make the following assumptions:

1. The two separation steps of the gel electrophoresis are independent. Therefore the distance between spots and hence pixels should reflect this characteristic. This makes the L_1 metric most appropriate in our application

2. Very long distances between corresponding spots in the two images are unlikely. So a first approach should be to find a solution there the sum of distances of corresponding spots is minimized.

3. During the protein separation no inversions should occur in the first or the second dimension. In practice this is typically true in the first dimension but sometimes might be violated in the second dimension due to local irregularities in the gel substance which causes proteins to travel at different speeds at different places of the gel. Therefore a second optimization goal should be to find a solution with a low number of inversions, especially on a local scale.

The well-known *Earth Mover's Distance* is able to capture the conditions 1. and 2., while for the third assumption further algorithmic care is needed.

Related Work

The earth mover's distance was proposed in [5], but there and in the subsequent papers it was employed as a pure *distance measure*. The actual transformations that lead to the distance were not really used

*Decodon GmbH, D-17489 Greifswald, bernhardt@decodon.com

†Institut für Formale Methoden der Informatik, Universität Stuttgart, D-70569 Stuttgart,{stefan.funke, sabine.storandt}@fmi.uni-stuttgart.de

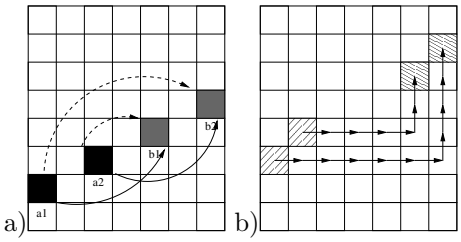


Figure 2: Shortcomings of simple EMD formulations.

or even derived. For the gel matching problem there are several approaches like [3] or [1] which fundametally differ from ours, though, as they first perform a spot detection and then a try to match corresponding spots. Hence they are more susceptible to the employed mathematical spot model.

2 Computation and Reconstruction of EMD Flows

2.1 Naive EMD Computation

EMD We are given two sets of points $A, B \subset \mathbb{R}^2$ and a weight function $w : A, B \to \mathbb{R}$. The term *Earth Mover's Distance* as coined by Rubner et al. [5] already captures its intuitive meaning. We consider the weight of points in set $A \subset \mathbb{R}^2$ as mass that needs to be moved to the locations given by set $B \subset \mathbb{R}^2$. We can compute the EMD via the following linear program:

$$\begin{aligned} min \quad & \sum_{a\in A, b\in B} f_{ab} d(a,b) && (1)\\ \forall a \in A \quad & \textstyle\sum_{b'\in B} f_{ab'} = w(a)\\ \forall b \in B \quad & \textstyle\sum_{a'\in A} f_{a'b} = w(b)\\ \forall a \in A, b \in B \quad & f_{ab} \geq 0\\ \sum_{a\in A, b\in B} f_{ab} \quad & = \min(\textstyle\sum_{a\in A} w(a), \sum_{b\in B} w(b)) \end{aligned}$$

Here, $f_{ab} = c$ denotes that c units of mass are moved from $a \in A$ to the location $b \in B$. From now on we refer to them as 'flows'. Moreover $d(a, b)$ describes the 'cost' of moving one unit of mass from a to b, in our case $d(a, b) = ||a - b||_1 = \sum_{i=1}^{n} |a_i - b_i|$. $w(a)/w(b)$ denotes the amount of supply/demand of mass at location $a \in A$, $b \in B$, respectively.

In general there are several optimal solutions of the EMD-LP. Note, that this is not specific for the L_1 metric but can also occur then using other distances like the euclidean metric, as depicted in figure 2 a). Here, the bold assignment would be the most reasonable one in our scenario, but the dashed correspondences lead to the same objective value under the L_1 as well as the L_2 metric. To get the EMD value, defined as $EMD(A, B) = (\sum_{a\in A} \sum_{b\in B} f_{ab} d(a, b))/(\sum_{a\in A, b\in B} f_{ab})$, this is not of any interest. But our goal now is to identify the solution that fits best to for our envisioned application scenario.

For sake of simplicity we restrict in the following to the case of *binary* images. Our proposed algorithm also generalizes in a straightforward manner to grayscale images (with mass > 1 at a single pixel), the proof for the running time gets more involved, though, introducing a dependency on the total mass.

We define the *Earth Mover's Corresponding Problem (EMCP)* as follows:

Definition 1 EMCP: *Given two sets $A, B \subset \mathbb{R}^2$. Compute the/a function $\phi : A \to B$ that minimizes the number of local inversions while maintaining an optimal solution for the corresponding EMD-LP.*

Essentially the EMCP is the problem of computing a specific *matching* in a geometric bipartite graph.

2.2 A Compact EMD Formulation for L_1

The size of the naive EMD formulation is essentially $\Theta(|A| \cdot |B|)$ which makes this formulation rather useless for real world instances from our application domain, e.g. for gel scans of around 1 Megapixel each with around 15% 'mass' we would already deal with more than 22 billion variables. In case of the L_1 metric, there is a simple and considerably more compact formulation[1]. Let us assume that our points in sets A and B have integer coordinates (like the pixel coordinates of two images). The idea is to decompose a flow from (x, y) to (x', y') with $x \leq x', y \leq y'$ into a sequence of $|x - x'|$ horizontal 'miniflows' $(x, y) \to (x + 1, y) \to (x + 2, y) \to \cdots \to (x', y)$ followed by a sequence of $|y - y'|$ vertical miniflows $(x', y) \to (x', y + 1) \ldots$. The respective LP formulation now has the following structure:

$$min \sum f_{xyx'y'}$$

$$\begin{aligned} a_{xy} + f^h_{(x-1)yxy} - f^h_{xy(x+1)y} - f^h_{xy(x-1)y} + f^h_{(x+1)yxy} &= E_{xy}\\ E_{xy} + f^v_{x(y-1)xy} - f^v_{xyx(y+1)} - f^v_{xyx(y-1)} + f^v_{x(y+1)xy} &= b_{xy}\\ E_{xy} &\geq 0\\ f_{xyx'y'} &\geq 0 \end{aligned}$$

Here, a_{xy} denotes the mass of point set A at position (x, y), the same for b_{xy}, respectively. $f^{h/v}_{xyx'y'}$ denotes the horizontal/vertical flow from pixel (x, y) to a (neighboring) pixel (x', y') (which now can take any positive value). E_{xy} can be interpreted as the intermediate distribution of mass after the horizontal shifts. Any feasible solution to the original formulation can be translated into this more compact formulation and vice versa as we will see in the following. The size of this formulation is *linear* in the size of the gel scan and its solution can be found using some standard LP solver like CPLEX [2] or specialized network

[1] This formulation has probably been used before but we could not come up with a reference.

simplex implementations like mcf [4]. Note that it is easy to handle sets/images with different masses by always requiring A to be the smaller set and replacing the second equality by an inequality.

2.3 Preliminary Flow Reconstruction

Having computed an optimal solution to the compact LP, our first goal is to construct actual flows of mass from set A to set B. We will proceed in two steps: first the horizontal miniflows are turned into horizontal flows, then, in a second step vertical flows are constructed from the intermediate mass distribution and the miniflows. Finally, in a last step, the horizontal and vertical flows are combined to flows for the original problem formulation.

1-dimensional Reconstruction: Essentially we first solve a 1-dimensional EMCP in each row of the image. All horizontal flows, the intermediate mass distributions and hence the excesses and deficits are known. We then sweep from left to right, balancing excesses and deficits in a first-in-first-out fashion. It is easy to see that this procedure balances the deficits and excesses at minimal cost, solving a 1-dimensional EMCP. Exactly the same procedure is applied vertically in each column to distribute the mass particles from the intermediate mass distribution to their final locations according to set B. It remains to combine the horizontal and vertical flows to obtain an actual matching between pixels in the source and the target image.

Gluing together Horizontal and Vertical Flows: At this point we can arbitrarily combine horizontal and vertical flows to obtain a cost optimal matching. In our implementation we employ a simple greedy strategy which combines incoming horizontal and outgoing vertical flows to diagonal flows.

While the resulting matching always yields flows which are optimal wrt the Earth Mover's distance, often this reconstruction does not lead to the 'correct' flows in our application. The problem is that the miniflows we get from a solution to LP (2.2) might not even allow for a reconstruction of the correct flows, see Figure 2 b).

2.4 Local Search on Preliminary Flows

Now we want to solve the *EMCP* based on our EMD solution. For real instances of our application we can not expect that a completely inversion-free solution exists. Nevertheless as a sanity check and for theoretical soundness it is desirable that our algorithm recovers a inversion free matching if such exists. We refer to sets allowing an inversion free matching from now on as *well-sorted*.

Definition 2 (well-sortedness) *We call two sets of points $A, B \subset \mathbb{R}^2$ with $|A| = |B|$* well-sorted *if there exists a bijective function/transformation/matching $\phi : A \to B$ with $\cup_{a\in A}\phi(a) = B$ and $\forall a, a' \in A$*

$$a_1 \lessgtr a'_1 \Rightarrow \phi(a)_1 \lessgtr \phi(a')_1 \quad \textit{and}$$
$$a_2 \lessgtr a'_2 \Rightarrow \phi(a)_2 \lessgtr \phi(a')_2$$

We note that for well-sorted sets A and B, the transformation ϕ that we want to recover and which witnesses the well-sortedness has optimal cost, i.e. has cost equal to the Earth Mover's distance (this actually requires proof).

After glueing together the miniflows we obtain a set of flows which are typically not well-sorted. Our idea now is to consider flows pairwise and exchange targets in case this improves 'well-sortedness'. Interestingly we can show that if the two sets A and B are well-sorted this leads to a well-sorted matching after $O(n^2)$ iterations. But even if A and B were not well-sorted, this procedure terminates and yields a drastically improved assignment. We denote by $y(f)$ for a flow $f = (a, b)$ its extent in y direction.

Definition 3 (Switch) *Let $f_1 = (a, b)$ and $f_2 = (a', b')$ be a pair of flows and $f_1^s = (a, b'), f_2^s = (a', b)$ their equivalents after switching the targets. A switch is feasible if*

$$\begin{aligned}
&1.)\ cost(f_1) + cost(f_2) = cost(f_1^s) + cost(f_2^s) \\
&2.a)\ \max(y(f_1), y(f_2)) > \max(y(f_1^s), y(f_2^s)) \qquad \textit{or} \\
&2.b)\ \max(y(f_1), y(f_2)) = \max(y(f_1^s), y(f_2^s)) \qquad \textit{and} \\
&\quad\ \max(x(f_1), x(f_2)) > \max(x(f_1^s), x(f_2^s))
\end{aligned}$$

We call case 2.a) a *height decreasing*, case 2.b) a *width decreasing* switch.

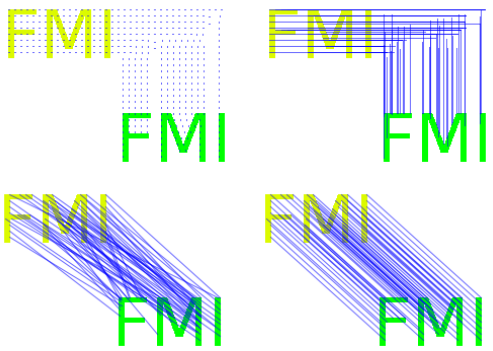


Figure 3: Reconstruction of a global translation: LP-solution in miniflows, 1-dimensional flows, greedy diagonal flows, after local search.

2.4.1 Algorithm

It is convenient to ensure that all y-coordinates are different and in particular, all heights of flows are pairwise different. We ensure this by standard techniques for symbolic perturbation on the y-coordinates. Our basic two-phase local search algorithm is then simple to state:

1. perform height decreasing switches between flows f_a and f_b as long as possible
2. perform width decreasing switches between flows f_a and f_b as long as possible

2.4.2 Runtime and Correctness

The running time of our switching algorithm crucially depends on the order in which we perform switches. For the height decreasing switches, we order the flows according to the y-coordinate of their sources and always check for the 'smallest' flow according to this order and switch it with the next smallest switchable flow. In the same manner we perform the width decreasing switches (ordered by x-coordinate).

Lemma 1 *For any a, a', flows associated with a and a' get switched at most once in phase 1 and at most once in phase 2 if we always use above criterion for selecting the next switch pair.*

This Lemma immediately yields a bound of $O(n^2)$ on the number of switches. A naive implementation can process one switch in time $O(n)$. For correctness, two core Lemmas have to be proven:

Lemma 2 *Let ϕ^* be a transformation witnessing the well-sortedness of sets A and B. Then ϕ^* does not allow any height or width decreasing switches.*

Lemma 3 *Let ϕ be the current set of flows, ϕ not witnessing well-sortedness of well-sorted sets A and B, then there exists a height or width decreasing switch.*

Particularly proving the latter Lemma is involved and requires a a closer examination of the structure of flows not witnessing well-sortedness. Using these Lemmas we obtain the our main theorem:

Theorem 4 *For well-sorted sets A and B our algorithm which starts with a cost-optimal solution terminates after $O(n^3)$ time and recovers a matching ϕ which witnesses well-sortedness of A and B.*

For not perfectly well-sorted instances correctness seems hard to formalize. From a biological point of view the solution of the EMCP is meaningful if we can assure well-sortedness at least for flows that are close to each other. It is not clear whether a further exploration of formal models for the 'correct' solution is worthwhile; as usual, the precise formalization of real-world problems seems challenging.

3 Experiments

In Figure 3 we have depicted the main steps of our approach. We compared the results of our algorithm to other approaches (like weighted bipartite matching) on model data. We always achieved the solution with the minimal number of inversions as well as the highest rate of correct correspondencies. Furthermore we checked on real data if our solution comes close to the so called manual edits (see figure 4), that could be seen as ground truth. Although we worked only on the binary versions of the image scans, we reconstructed about 84% (average) of the flows correctly.

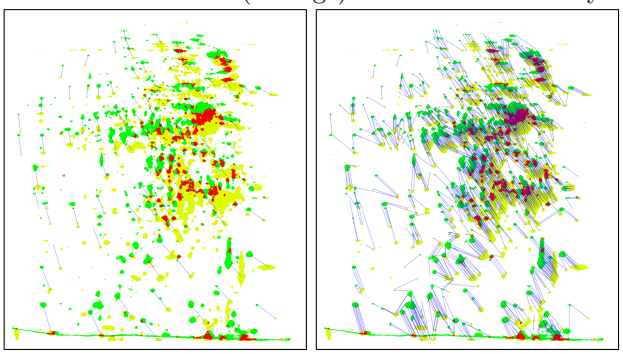


Figure 4: Comparison of a manual edit (left) and the flows resulting from our algorithm (middle, subsampled). Green: Image A, Yellow: Image B, Red:Overlap.

4 Conclusion

We presented an algorithm which faithfully determines correspondencies between protein spots in our concrete application from computational biology. Due to the nature of the physical process, those problem instances are typically not well-sorted but somewhat 'close' to being well-sorted. While truly well-sorted instances could be trivially solved by a simple sorting procedure, our algorithm also works for those real-world instances – the sorting approach does not. For well-sorted instances, we can prove our algorithm to be correct. Other application domains where this kind of matching could be of interest are in the field of vision or video compression.

References

[1] M. Berth, F. Moser, M. Kolbe, and J. Bernhardt. The state of the art in the analysis of 2-dimensional gel electrophoresis images. *Appl Microbiol Biotechnol*, 2007.

[2] R. E. Bixby. Implementing the simplex method: The initial basis. *INFORMS Journal on Computing*, 4(3):267–284, 1992.

[3] A. Efrat, F. Hoffmann, K. Kriegel, C. Schultz, and C. Wenk. Geometric algorithms for the analysis of 2d-electrophoresis gels. *Journal of Computational Biology*, 9(2):299–315, 2002.

[4] A. Loebel. Mcf version 1.3 - a network simplex implementation. available for academic use free of charge. `http://www.zib.de`, 2004.

[5] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, 2000.

On some connection problems in straight-line segment arrangements*

Helmut Alt[†‡] Sergio Cabello[§] Panos Giannopoulos[†‡] Christian Knauer[¶]

Abstract

We study the complexity of some problems of the following type: Given a set of straight-line segments in the plane and a set of cells in the induced arrangement, compute the minimum number of segments one needs to remove so that the cells become connected. We show that the problems of connecting two cells and connecting all cells are both NP-hard. We also discuss several polynomial-time solvable and fixed-parameter tractable cases.

1 Introduction

Let S be a set of straight-line segments in $\mathbb{R}^2$ and $\mathcal{A}(S)$ be the arrangement induced by S.

In the 2-Cells-Connection problem we are given two points $s, t \in \mathbb{R}^2$ and we want to compute a set $S' \subseteq S$ of minimum possible size, with the property that s and t belong to the same cell of $\mathcal{A}(S \setminus S')$. In other words, we want to compute an s-t path that *crosses* the minimum number of segments of S counted without multiplicities. The *cost* of a path is the total number of segments crossed by it.

In the All-Cells-Connection problem we want to compute a set $S' \subseteq S$ of minimum possible size such that $\mathcal{A}(S \setminus S')$ consists of one cell only.

Without loss of generality, we assume that every segment in S intersects at least two other segments and that both endpoints of a segment are intersection points. We say that two segments cross if and only if they intersect at a common interior point (a segment crossing).

Results. We show that 2-Cells-Connection is NP-hard even when no three segments intersect at a point. When any three segments may intersect only at a common endpoint, the problem is fixed-parameter tractable with respect to the number of segment crossings. We also consider an application variant of the problem where the segments lie inside a polygon with holes P and have their endpoints on its boundary and the s-t path must also stay inside P, and show it to be fixed-parameter tractable with respect to the number of holes of P. On the other hand, we show that All-Cells-Connection is NP-hard even if no three segments intersect at a point and there are no segment crossings.

Related work. Bereg and Kirkpatrick [1] studied the problem of computing the so called barrier resilience in wireless sensor networks, i.e., the minimum number of disks whose removal connects two given cells in an arrangement of unit disks (sensors), and gave a 3-approximation algorithm based on a shortest path computation in the dual of the arrangement. The complexity of the problem though remains open.

2 Connecting two cells

We show that 2-Cells-Connection is NP-hard by a reduction from MAX-2-SAT, a well studied NP-complete problem [6]: Given a propositional CNF formula Φ with m clauses on n variables and at most two variables per clause, decide whether there exists a truth assignment that satisfies at least k clauses, for some given $k \in \mathbb{N}$, $k \leq m$. Let ℓ be the maximum number of occurrences of a variable in Φ. For simplicity we can assume that $\ell \leq 3$ since the restricted version of MAX-2-SAT where any variable occurs in at most 3 clauses is still NP-complete [9].

Using Φ we construct an instance consisting of a set of segments S and two points s and t as follows.

Abusing the terminology slightly, by a *segment* we actually mean a set of identical single segments stacked on top of each other. The cardinality of the set is the *weight* of the segment. Either all or none of the single segments in the set can by crossed by a path.

There are three different types of segments, T_1, T_2, and T_3, according to their weight, see Fig. 1. Segments of type T_1 have weight 1 (single or light segments), while segments of type T_2, T_3 have weight $(m+1)$ and $18n(m+1)$ respectively (heavy segments).

First, we construct a polygon, called the *tunnel*, with heavy boundary segments of type T_3, see Fig. 1. There are 21 boundary segments in total. The tunnel has a 'zig-zag' shape and can be seen as having 8 corridors, $C_1, \ldots, C_8$. It starts with C_1, the *main* corri-

*Part of this research was conducted at the 9th McGill - INRIA Barbados Workshop on Computational Geometry, 2010.

[†]Institut für Informatik, Freie Universität Berlin, Takustraße 9, D-14195 Berlin, Germany, {alt, panos}@mi.fu-berlin.de

[‡]This research was supported by the German Science Foundation (DFG) under grant Kn 591/3-1.

[§]Department of Mathematics, IMFM and FMF, University of Ljubljana, Jadranska 19, SI-1000 Ljubljana, Slovenia, sergio.cabello@fmf.uni-lj.si

[¶]Institut für Informatik, Universität Bayreuth Universitätsstraße 30 D-95447 Bayreuth Germany, christian.knauer@uni-bayreuth.de

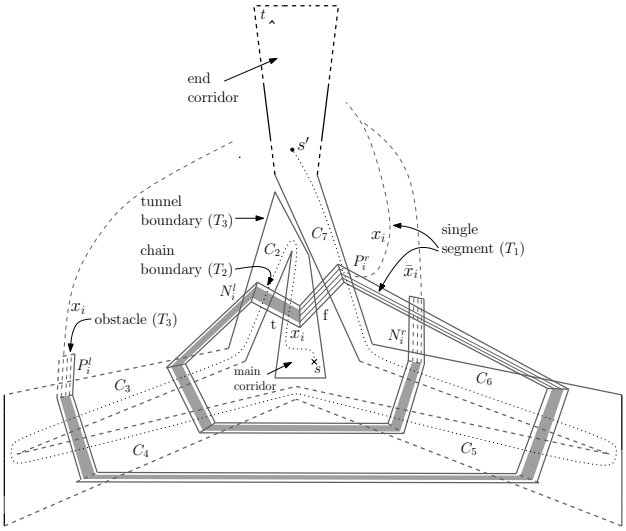


Figure 1: Variable chain and a possible path from s to s'; a shaded area represents a set of ℓ single segments.

dor (at the center of the figure), which contains point s, then it turns left to C_2, then right, etc., gradually turning around to C_7 and then to the *end* corridor C_8 (at the top). The latter contains point t. The total size of the tunnel is $21 \cdot 18n(m+1) = \mathcal{O}(nm)$.

The rest of the construction will force any s-t path of some particular cost (to be given shortly) to stay always in the interior of the tunnel.

Each variable of Φ is represented by a *chain* of segments, see Fig.1. The chain has 12 pieces and each piece is separated from its neighbors by (or ends at, when it is the last one) a 'short' heavy segment of type T_3, referred to as an *obstacle*; with the exception of the two ends of the chain, all obstacles lie in the interior of the tunnel. Each piece consists of:

(i) 2 *chain boundary* segments of type T_2; every endpoint of such a segment lies on an obstacle.
(ii) ℓ single segments (type T_1).

It has $2(m+1)+\ell$ segments in total and it is said to be crossed by a path if and only if all of its segments are crossed.

Consider the example of the variable chain x_i in Fig. 1, and its part in the main corridor. From the obstacle, the chain extends to the left towards corridor C_2 and to the right towards corridor C_7. Consider the right half of the chain. In the first piece all ℓ single segments lie between the two boundary segments: they start at the obstacle in the main corridor and end at the obstacle in C_7. In the second piece, denoted by P_i^r (positive-right), all single segments start at this latter obstacle. Some (in the example just one) end inside the end corridor at intersection points that represent clauses with a positive occurrence of x_i; in Fig. 1, these segments are drawn by dashed lines, which can be made straight by stretching the

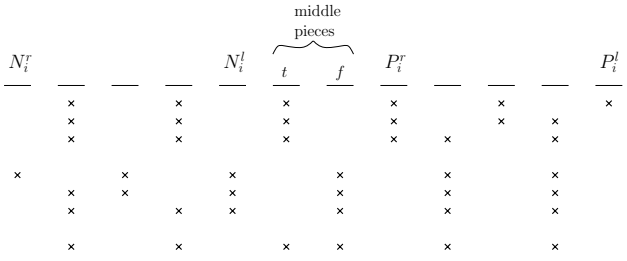


Figure 2: All possible combinations of six pieces of a variable chain crossed by a minimum-cost s-s' path.

end corridor sufficiently. The rest continue down to C_6, always lying between the boundary segments of the piece, and end at an obstacle inside. The third, fourth, and fifth piece are similar to the first one: all single segments of a piece (represented by a shaded area) lie between its respective boundary segments. The chain makes three consecutive right turns, from C_6 to C_5, then to C_4, and then to C_3. In the last piece, P_i^l (positive-left), all single segments start at an obstacle in C_3. Again, some go up to the end corridor, to clauses with a positive occurrence of x_i, while the rest end at an obstacle outside the tunnel. Note that any single segment may intersect a chain boundary segment only outside the tunnel. The left half of the chain is constructed in an analogous fashion. Its second and sixth piece are denoted by N_i^l (negative-left) and N_i^r (negative-right) respectively; some single segments coming from these pieces go to clauses with a negative occurrence of x_i.

Each clause of Φ is represented by an intersection of two single segments inside the end corridor; see Fig. 3 for an example of the overall construction. Each segment corresponds to some literal x_i or $\bar{x}_i$ in the clause: in the first case the segment comes from either P_i^r or P_i^l, while in the second one it comes from either N_i^r or N_i^l. For the construction, these choices for a each clause can be made arbitrarily, provided that one segment intersects the tunnel from the left side and the other one from the right. In this way, the end corridor is obstructed by m pairs of intersecting segments such that any path from the intermediate point s' to point t staying inside the tunnel must intersect at least one segment from each pair.

Observe that any minimum-cost path from s to s' that stays in the interior of the tunnel crosses only 6 pieces from each variable chain (and no obstacles), where at least one piece from every two consecutive ones is crossed; see Fig. 1 for an example of such a path. There are 7 such possible sets of pieces, see Fig. 2, and the choice is made independently for each chain. Consider the two *middle* pieces, parts of which lie in the main corridor. When only the left one is crossed (first three sets), P_i^r is crossed as well, while none of N_i^l, N_i^r is crossed: effectively, x_i is set to *true*.

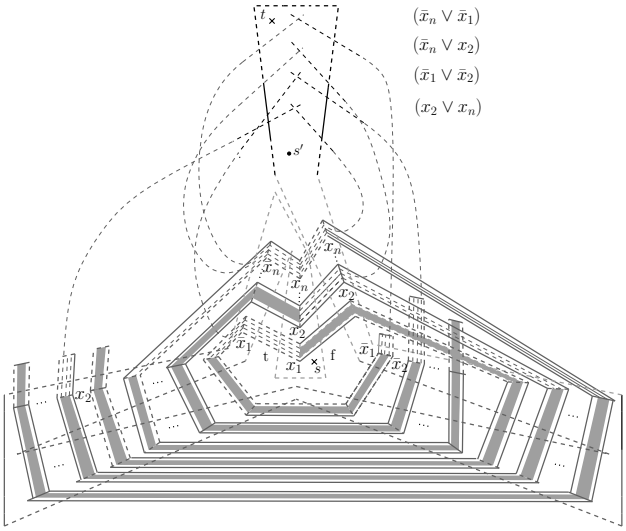


Figure 3: Example of overall construction.

Symmetrically, when only the right piece is crossed (next three sets), x_i is set to *false*. The last set does not correspond to a valid truth assignment. The total cost for such a path is $6n(2(m+1)+\ell)$.

Thus, for an s-t path to cost $6n(2(m+1)+\ell)+k$, for some $0 \leq k < m$, its subpath from s' to t in the end corridor must cross at most k single segments that have not been crossed before. This implies that there are at least $(m-k)$ clauses from each of which at least one segment has been already crossed by the s-s' subpath. For such a clause, let $u_i \in \{x_i, \bar{x}_i\}$ be the literal that the crossed segment corresponds to. By construction, if $u_i = x_i$, at least one of P_i^r, P_i^l has been crossed, while if $u_i = \bar{x}_i$, at least one of N_i^r, N_i^l has been crossed. From the discussion above, x_i is set to true in the first case and to false in the second one, and, hence, the clause is satisfied.

Lemma 1 *There exists an s-t path with a cost of at most $6n(2(m+1)+\ell)+k$ if and only if there exists a truth assignment that satisfies at least $(m-k)$ clauses of Φ.*

We can modify our construction by replacing every heavy segment with a set of distinct parallel single segments in a way such that every single segment in S intersecting the original heavy segment now intersects all the segments in the new set and making sure that no three segments have a point in common.

Theorem 2 2-Cells-Connection *is NP-hard even when no three segments intersect at a point.*

We can reduce 2-Cells-Connection to the minimum color path problem (MCP): Given a graph G with colored (or labeled) edges and two of its vertices, find a path between the vertices that uses the minimum possible number of colors. We color the edges of the dual graph G of $\mathcal{A}(S)$ as follows: two edges of G get the same color if and only if their corresponding edges in $\mathcal{A}(S)$ lie on the same segment of S. Then, finding an s-t path of cost k in $\mathcal{A}(S)$ amounts to finding a k-color path in G between its two vertices that correspond to the cells which s, t lie in.

However, MCP is NP-hard [2] and W[1]-hard [5] (with respect to the number of colors in the path) even for planar graphs, it has a polynomial-time $\mathcal{O}(\sqrt{n})$-approximation algorithm and is non-approximable within any polylogarithmic factor [8].

2.1 Tractable cases

Consider the colored dual graph G of $\mathcal{A}(S)$ as defined above. A face of G (except the outer one) corresponds to a point of intersection of some $r \geq 2$ segments and has r colors and, depending on the type of intersection, from r to $2r$ edges. For example, for $r = 2$ we can get two multiple edges, a triangle, or a quadrilateral, with two distinct colors.

When any three segments may intersect only at a common endpoint and no two segments cross, G can only have multiple edges, bi-chromatic triangles, and arbitrary large faces where all edges have different colors. In this case, since two segments can intersect only at one point, each color induces a connected subgraph of G, in fact a tree for there can be no monochromatic cycle in G. Then, 2-Cells-Connection reduces to a shortest path problem in an uncolored modification of G, where each monochromatic tree is contracted into a star.

Generalizing this, if we allow k segment crossings, i.e., bi-chromatic quadrilaterals in G, we can easily find a minimum-cost s-t path as follows. Let $C \subseteq S$ be the set of the (at most $2k$) segments participating in these crossings. For every possible subset C' of C, first contract every edge of G corresponding to a segment in C', and then assign an infinite weight to every edge corresponding to a segment in $C \setminus C'$. For a fixed subset, a solution can be found by computing a shortest path in an uncolored weighted graph.

Theorem 3 2-Cells-Connection *is fixed-parameter tractable with respect to the number of segment crossings if any three segments may intersect only at a common endpoint.*

2.2 An application

Let P be a polygon with h holes and S be a set of n segments lying inside P with their endpoints on its boundary; see Fig. 4, where, for clarity, the boundary of P is drawn by a set of simple closed curves. We consider the restricted 2-Cells-Connection problem on $P \cup S$ where the s-t path may not cross the boundary. This version is also NP-hard by a simple reduction from the general one.

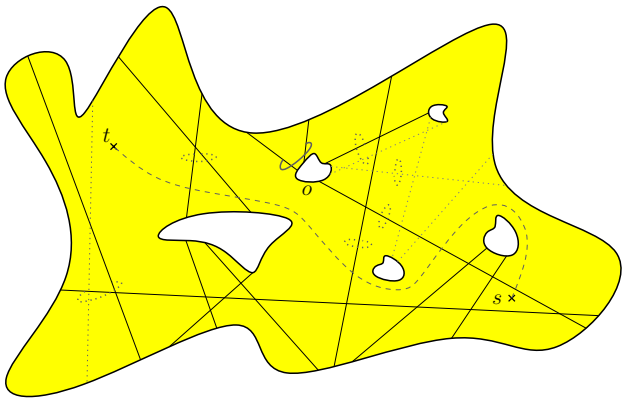


Figure 4: Some segment clusters in a polygon with holes and a minimum-cost s-t path.

We partition S into clusters using homotopies as follows: two segments belong in the same cluster if and only if there is a continuous transformation from one to the other, during which the endpoints stay on the boundary of P; for this, s and t are treated as special holes. In Fig. 4 for example, the segments that touch hole o give four clusters. Using simple topological arguments we prove the following:

Lemma 4 *S can be partitioned into $O(h^4)$ clusters with the property that either all or none of the segments in a cluster are crossed by a minimum-cost s-t path.*

A minimum-cost s-t path can now be easily found by testing all possible subsets of clusters.

Theorem 5 *The restricted* 2-Cells-Connection *problem in a polygon with holes is fixed-parameter tractable with respect to the number of holes.*

3 Connecting all cells

We prove that All-Cells-Connection is NP-hard by a reduction from the well-known NP-hard problem of feedback vertex set (FVS) in planar graphs [6]: Given a planar graph G, find a minimum-size set of vertices X such that $G - X$ is acyclic.

First, we subdivide every edge of G obtaining a planar bipartite graph G'. It is clear that G' has a feedback vertex set of size k if and only if G has one. Next, we use the result by de Fraysseix et al. [4] (see also Hartman et al. [7]), which states that every planar bipartite graph is the intersection graph of horizontal and vertical segments, where no two of them cross. Let S be the set of segments whose intersection graph is G'; it can be constructed in polynomial time. Since G' has no triangles, no three segments of S intersect at a point. Finally, observe that all cells in $\mathcal{A}(S)$ become connected by removing k segments if and only if G' has a feedback vertex set of size k.

Theorem 6 All-Cells-Connection *in NP-hard even if no three segments intersect at a point and there are no segment crossings.*

It is also easy to see that a k-size solution to All-Cells-Connection corresponds to a k-size solution of FVS in the intersection graph of the input segments. For general graphs, FVS is fixed-parameter tractable when parameterized with the size of the solution [3], and has a polynomial-time 2-approximation algorithm [10].

Corollary 7 All-Cells-Connection *is fixed-parameter tractable with respect to the size of the solution and has a polynomial-time 2-approximation algorithm.*

Acknowledgments

We would like to thank Primož Škraba for bringing to our attention some of the problems studied in this abstract.

References

[1] S. Bereg and D. G. Kirkpatrick. Approximating barrier resilience in wireless sensor networks. In *Proc. of the 5th ALGOSENSORS*, volume 5804 of *LNCS*, pages 29–40. Springer, 2009.

[2] H. Broersma, X. Li, G. Woeginger, and S. Zhang. Paths and cycles in colored graphs, p.299. *Australasian J. Combin.*, 31:299–312, 2005.

[3] J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.*, 74:1188–1198, 2008.

[4] H. de Fraysseix, P. O. de Mendez, and J. Pach. Representation of planar graphs by segments. *Intuitive Geometry*, 63:109–117, 1991.

[5] M. R. Fellows, J. Guo, and I. Iyad. The parameterized complexity of some minimum label problems. *J. Comput. Syst. Sci.*, 76:727–740, 2010.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[7] I. B. Hartman, I. Newman, and R. Ziv. On grid intersection graphs. *Discrete Mathematics*, 87:41–52, 1991.

[8] R. Hassin, J. Monnot, and D. Segev. Approximation algorithms and hardness results for labeled connectivity problems. *J. Comb. Optim.*, 14(4):437–453, 2007.

[9] V. Raman, B. Ravikumar, and S. S. Rao. A simplified NP-complete MAXSAT problem. *IPL*, 65(1):1–6, 1998.

[10] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

A conjecture about an upper bound of the RMSD between linear chains

Christian L. Müller, Ivo F. Sbalzarini *

Abstract

We combine stochastic global optimization and analytical geometry in order to conjecture an upper bound for the Root Mean Square Deviation (RMSD) between linear chains of N beads with link length b after optimal roto-translational fitting. We report pairs of putative extremal configurations and an analytical expression for the RMSD between them, asymptotically approaching $\frac{1}{4}\sqrt{\frac{5}{3}}bN$ for large N.

1 Introduction

Since the pioneering works of Flory [3] chain models have been instrumental for theoretical studies of polymers. The simplest models are linear chains such as the freely jointed (or ideal) Random Walk (RW) or the Self-Avoiding Walk (SAW). These models provide the theoretical basis for more complex (bio-)polymer and protein models [2]. A linear chain is a configuration of ordered points (beads, atoms) in three-dimensional space, where the Euclidean distance between consecutive beads is constrained to an arbitrary but fixed constant b, the bond or link length.

The advent of efficient algorithms for determining the minimum Root Mean Square Deviation (RMSD) [10, 7] between two linear chains has triggered research in characterizing the configuration space of chain ensembles using RMSD as the standard distance metric in the field. Starting from ideal RW ensembles [11] the analysis has been extended to more complex polymer and protein models [14].

While the minimum RMSD between two configurations reaches a trivial lower bound of 0 for identical chains, a tight upper bound – or the two configurations of linear chains that are most dissimilar from each other – is still unknown.

We address this problem using a combination of global optimization and analytical geometry. We numerically determine the maximum RMSD of RW chains for several N and deduce from these results a general formula for odd N. We conjecture that the asymptotic limit of this formula is valid for all N and that it is an upper bound for the maximum RMSD between general linear chains.

*Institute of Theoretical Computer Science, Department of Computer Science, ETH Zürich, and Swiss Institute of Bioinformatics, christian.mueller@inf.ethz.ch, ivos@ethz.ch

2 Definitions and Methods

2.1 The minimum RMSD

We represent two configurations of N beads each by the matrices $X, Y \in \mathbb{R}^{3\times N}$. Each column in X, Y is denoted $\mathbf{x}^{(i)}, \mathbf{y}^{(i)}$ and contains the three-dimensional Cartesian coordinates of the i^{th} bead of the configuration. In a linear chain model, consecutive beads are connected by links of fixed length b. Calculating the minimum RMSD $D(X,Y)$ between X and Y comprises two steps: (i) translating the centers of mass $\mathbf{x}_{\text{cm}}$ and $\mathbf{y}_{\text{cm}}$ of both configurations to the origin, leading to repositioned chains X_0 and Y_0 with columns $\mathbf{x}_0^{(i)}, \mathbf{y}_0^{(i)}$; (ii) determining the optimal rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times 3}$, such that:

$$D^2(X,Y) \doteq \min_{\mathbf{R}} \frac{1}{N}\|\mathbf{R}X_0 - Y_0\|_2^2 \,. \tag{1}$$

The optimal rotation matrix $\mathbf{R}$ can be determined using Singular Value Decomposition (SVD) [6, 7] or quaternions [8]. It is a special case of the *orthogonal Procrustes problem* ([4], pp. 601) where $\mathbf{R}^{\text{T}}\mathbf{R} = \mathbf{I}_3$ (the 3×3 identity matrix) and $\det \mathbf{R} = 1$.

$D^2(X,Y)$ can be expressed in terms of the radii of gyration of X and Y, $R_G(X)$ and $R_G(Y)$, as [10, 11]:

$$D^2(X,Y) = R_G^2(X) + R_G^2(Y) - 2\frac{1}{N}\sum_{i=1}^{N} \tilde{\mathbf{x}}_0^{(i)} \cdot \mathbf{y}_0^{(i)} \tag{2}$$

with $\tilde{\mathbf{x}}_0^{(i)} = \mathbf{R}\mathbf{x}_0^{(i)}$ and $R_G^2(X) = \text{tr}(X^{\text{T}}X)$. The term $\frac{1}{N}\sum_{i=1}^{N} \tilde{\mathbf{x}}_0^{(i)} \cdot \mathbf{y}_0^{(i)}$ describes the structural correlation between X and Y after optimal superposition and can be re-written as [1]:

$$\frac{1}{N}\sum_{i=1}^{N} \tilde{\mathbf{x}}_0^{(i)} \cdot \mathbf{y}_0^{(i)} = \frac{\sum_{i=1}^{N} \tilde{\mathbf{x}}_0^{(i)} \cdot \mathbf{y}_0^{(i)}}{\sqrt{\sum_{i=1}^{N} \mathbf{x}_0^{(i)2} \sum_{i=1}^{N} \mathbf{y}_0^{(i)2}}} R_G(X) R_G(Y) \,. \tag{3}$$

Betancourt and Skolnick [1] refer to the fraction in Eq. (3) as the *aligned correlation coefficient* $ACC(X,Y)$. The radius of gyration R_G of a chain X is roto-translation invariant and can be written as:

$$\begin{aligned} R_G^2(X) &= \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \mathbf{x}_{\text{cm}}\|_2^2 \\ &= -\mathbf{x}_{\text{cm}} \cdot \mathbf{x}_{\text{cm}} + \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}^{(i)}\|^2 \,. \end{aligned} \tag{4}$$

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

From Eq. (2), McLachlan derived relative lower and upper bounds for $D^2(X,Y)$ of two given chains X and Y [11]:

$$0 \leq D^2(X,Y) \leq R_G^2(X) + R_G^2(Y)\,. \tag{5}$$

2.2 The linear chain RW model

A linear chain X is represented in internal coordinates. We denote by θ_i the angle between three consecutive beads $\mathbf{x}^{(i)}$, $\mathbf{x}^{(i+1)}$, $\mathbf{x}^{(i+2)}$. The dihedral between the two consecutive planes spanned by ($\mathbf{x}^{(i)}$, $\mathbf{x}^{(i+1)}$, $\mathbf{x}^{(i+2)}$) and ($\mathbf{x}^{(i+1)}$, $\mathbf{x}^{(i+2)}$, $\mathbf{x}^{(i+3)}$) is ω_i (see Fig. 1a). A chain of N beads with fixed bond length

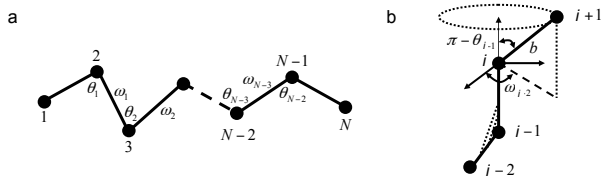


Figure 1: a. Definition of the angles θ_i and dihedrals ω_i characterizing an anchored walk of length N. b. Illustration of the trigonometric map $\mathbf{q}_X \to X$ from internal coordinates $\mathbf{q}_X$ to Cartesian coordinates X.

b has $N-2$ angles and $N-3$ dihedrals, resulting in $M = 2N-5$ degrees of freedom. It is described by the internal coordinate vector $\mathbf{q}_X \doteq \{\theta_i | i = 1 \ldots N-2, \omega_i | i = 1 \ldots N-3\}$. For an ideal RW chain, the direction of each link is chosen uniformly random on the unit sphere by sampling the $\cos(\theta_i)$ uniformly from $[0,1]$ and the ω_i uniformly from $[0,\pi]$ [2]. The link length between consecutive beads is fixed to b, the mass of each bead is $m = \frac{1}{N}$.

In order to avoid redundant chains that can be superimposed by rigid-body translation and rotation, we use "anchored" walks where $\mathbf{x}^{(1)}$ is placed at the origin, $\mathbf{x}^{(2)}$ along the x-axis at $(b,0,0)^{\mathrm{T}}$, and the link between $\mathbf{x}^{(2)}$ and $\mathbf{x}^{(3)}$ is contained in the xy-plane. This uniquely defines the overall position and orientation of the walk [13].

A pair of anchored RW chains (X,Y) of N beads each is represented by $\mathbf{q}_S = (\mathbf{q}_X, \mathbf{q}_Y)$. The transformation from internal coordinates to three-dimensional Cartesian coordinates is denoted by $J(\mathbf{q}_S) = (X,Y)$ (see Fig. 1b).

2.3 The maximum RMSD problem

The *maximum RMSD problem* (MAX-RMSD) is stated as a continuous, non-convex max–min optimization problem. We seek the specific pair of chains ($X^N_{\max}$, $Y^N_{\max}$) of N beads each that maximizes $D^2(X,Y)$ over all possible X and Y, hence:

$$\begin{aligned}(X^N_{\max}, Y^N_{\max}) &= \arg\max_{X,Y} D^2(X,Y) \\ &= \arg\max_{X,Y} \min_{\mathbf{R}} \frac{1}{N} \|\mathbf{R}X_0 - Y_0\|_2^2\,.\end{aligned} \tag{6}$$

We refer to the pair $(X^N_{\max}, Y^N_{\max})$ as the *extremal configurations* of the chain ensemble in the RMSD sense. If both X and Y are anchored linear RW chains, we call the problem RW-MAX-RMSD. Its $D_{\max}(N) = \sqrt{D^2(X^N_{\max}, Y^N_{\max})}$ is an upper bound for the maximum RMSD of all linear chain ensembles.

The inner minimization problem can be solved analytically by constructing the optimal rotation matrix $\mathbf{R}$ from SVD [6, 7, 4] or using quaternions [8, 9]. The distance constraints on the positions of consecutive beads $\|\mathbf{x}_i - \mathbf{x}_{i+1}\|_2 = b, \quad i = 1, \ldots, N-1$, are satisfied by using internal coordinates $\mathbf{q}$.

The outer maximization problem can be formulated as a constrained, non-convex black-box optimization problem in $n = 2(2N-5) = 4N-10$ dimensions. For convenience we consider the unit hypercube as feasible domain, i.e., candidate solution vectors $\hat{\mathbf{q}}_S$ are in $[0,1]^n$. The unique map $T : \hat{\mathbf{q}}_S \in [0,1]^n \to \mathbf{q}_S \in ([0,1]^{2(N-2)}, [0,\pi]^{2(N-3)})$ transforms any candidate solution vector to internal coordinates of a chain. The black-box objective function f to be maximized then reads:

$$\begin{aligned}f(\hat{\mathbf{q}}_S) \equiv D^2\left(J(T(\hat{\mathbf{q}}_S))\right) &= D^2(X,Y) \\ &= \min_{\mathbf{R}} \frac{1}{N} \|\mathbf{R}X_0 - Y_0\|^2\,.\end{aligned} \tag{7}$$

Note that this formulation is known *a priori* to become two-fold degenerate if two consecutive links in a trial configuration are co-linear. First, the corresponding dihedral angles are then undefined, i.e., the configuration remains the same regardless of their values. Second, the optimal rotation matrix has only rank 1, permitting infinitely many rotations that minimize $D^2(X,Y)$ [8].

3 Numerical optimization results

We numerically solve the RW-MAX-RMSD problem for pairs of configurations with $N = 3, \ldots, 16$ beads. The dimensionality of the problem is thus ranging from $n = 2, \ldots, 50$. We use two optimization algorithms: (i) Sequential Quadratic Programming (SQP) and (ii) Best Local Restart Covariance Matrix Adaptation Evolution Strategy (BLR-CMA-ES). For SQP, the MATLAB implementation *fmincon* is used. For box-constrained black-box optimization problems, this implementation uses an active-set SQP algorithm with approximate BFGS and line search. BLR-CMA-ES is a local restart variant of the variable-metric optimizer CMA-ES [5]. Details of

BLR-CMA-ES and the set-up of the numerical experiments have been described elsewhere [12].

The putative optimal solutions $(X^N_{\max}, Y^N_{\max})$ found by SQP and BLR-CMA-ES agree for $N = 3, 5, 7, 11$. For all other instances, BLR-CMA-ES consistently outperforms SQP, finding configurations with larger minimum RMSD than those found by SQP. These extremal configuration are shown in Fig. 2. For odd

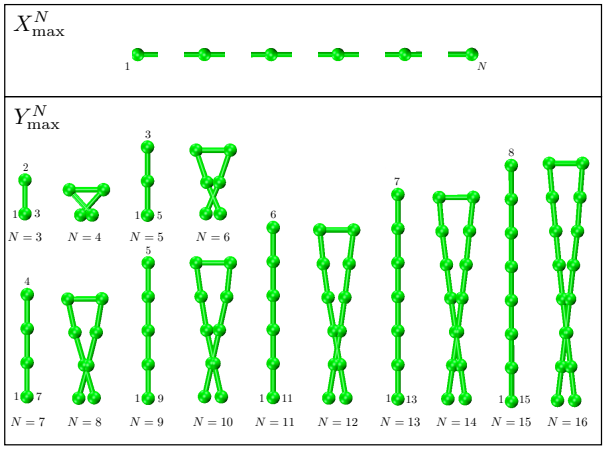


Figure 2: Extremal configurations $(X^N_{\max}, Y^N_{\max})$ of linear RW chains with $N = 3, \ldots, 16$ found by BLR-CMA-ES. The upper box shows the extended configurations $X^N_{\max}$. The lower box shows the corresponding configurations $Y^N_{\max}$. For odd N, $Y^N_{\max}$ is a linear rod of half the length with beads $\frac{N+3}{2}$ to N folded back onto beads $\frac{N-1}{2}$ to 1. For even N, $Y^N_{\max}$ is a planar hairpin where the links from beads $\frac{N+2}{2}$ to N cross the links from beads $\frac{N}{2}$ to 1.

N, they follow a regular geometric pattern: one configuration always is the fully extended linear rod, the other is a linear rod of half the length with beads $\frac{N+3}{2}$ to N folded back onto beads $\frac{N-1}{2}$ to 1. For even N, the first extremal configuration is again the fully extended linear rod, whereas the other is a *planar hairpin* with crossed ends. For odd N, the ACC of the extremal configurations is virtually 0 ($< 10^{-15}$), for even N it is $< 10^{-3}$. These optima found by BLR-CMA-ES suggest a near-linear dependence of $D_{\max}(N)$ on N with a best linear fit of $D_{\max}(N) \approx 0.3251bN - 0.04013$.

4 The MAX-RMSD conjecture

The above results suggest that the extremal configurations for odd N follow a simple geometric pattern: one configuration is the fully extended linear rod, the other one a linear fold-back of half the length.

Conjecture 1 *The fully extended linear rod and its linear fold-back configuration are the optimal solution of the RW-MAX-RMSD problem for all odd N.*

Under this assumption we derive a general formula for $D_{\max}(N)$ for odd N. Combining Eqs. (2) and (3) we find:

$$\begin{aligned} D^2_{\max}(N) &= D^2(X^N_{\max}, Y^N_{\max}) = R^2_G(X^N_{\max}) + R^2_G(Y^N_{\max}) \\ &- 2ACC(X^N_{\max}, Y^N_{\max}) R_G(X^N_{\max}) R_G(Y^N_{\max}) \\ &= R^2_G(X^N_{\max}) + R^2_G(Y^N_{\max}), \end{aligned} \tag{8}$$

provided that $ACC(X^N_{\max}, Y^N_{\max}) = 0$ for all odd N.

Lemma 1 *The $ACC(X^N_{max}, Y^N_{max})$ for odd N is 0.*

Proof. Without loss of generality we assume that $X^N_{\max}$ is the fully extended rod and $Y^N_{\max}$ the back-folded one, and that their centers of mass are at $(0, 0, 0)$. For odd N, the problem of optimal superposition then reduces to a rotation in the xy-plane. We define the x-axis to be aligned with $X^N_{\max}$ after optimal superposition. $Y^N_{\max}$ forms a certain rotation angle α with $X^N_{\max}$ as shown in Fig. 3. We show that

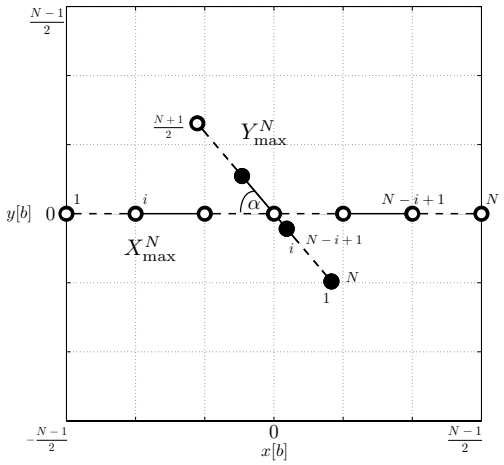


Figure 3: Calculation of the RMSD between $X^N_{\max}$ and $Y^N_{\max}$ for odd N after optimal superposition. $X^N_{\max}$ is the extended configuration and $Y^N_{\max}$ the folded one. Open circles (○) represent positions that are occupied by single beads, filled circles (●) indicate positions occupied by two beads. The two configurations enclose a planar angle α.

for the specific pair of configurations $(X^N_{\max}, Y^N_{\max})$ the $ACC(X^N_{\max}, Y^N_{\max})$ is 0 for any rotation angle α and any odd N by recalling the definition of ACC for two optimally aligned chains X, Y:

$$ACC(X, Y) = \frac{\sum_{i=1}^N \mathbf{x}^{(i)} \cdot \mathbf{y}^{(i)}}{\sqrt{\sum_{i=1}^N \mathbf{x}^{(i)2} \sum_{i=1}^N \mathbf{y}^{(i)2}}}. \tag{9}$$

The denominator of this expression is always positive since the two factors under the square root are sums over squared bead coordinates.

From Fig. 3 we see that the coordinate vectors $\mathbf{x}_{\max}^{(i)}$ only have non-zero entries in x-direction. Furthermore, the x coordinate of the i^{th} bead in $X_{\max}^N$ is the negative of the x coordinate of the $(N-i+1)^{\text{th}}$ bead. The central bead (i.e., the $\left(\frac{N+1}{2}\right)^{\text{th}}$ bead) in $X_{\max}^N$ is at $(0,0,0)$, so the scalar product with its corresponding bead in $Y_{\max}^N$ is 0. The positions of the i^{th} and $(N-i+1)^{\text{th}}$ beads in $Y_{\max}^N$ are identical (filled circles in Fig. 3) for all α. The numerator in Eq. (9) hence becomes:

$$\sum_{i=1}^{N} \mathbf{x}_{\max}^{N,(i)} \cdot \mathbf{y}_{\max}^{N,(i)} = \sum_{i=1}^{\frac{N+1}{2}-1} \mathbf{x}_{\max}^{N,(i)} \cdot \mathbf{y}_{\max}^{N,(i)} + 0 + \sum_{i=\frac{N+1}{2}-1}^{N} \mathbf{x}_{\max}^{N,(i)} \cdot \mathbf{y}_{\max}^{N,(i)} = -\sum_{i=\frac{N+1}{2}-1}^{N} \mathbf{x}_{\max}^{N,(i)} \cdot \mathbf{y}_{\max}^{N,(i)} + \sum_{i=\frac{N+1}{2}-1}^{N} \mathbf{x}_{\max}^{N,(i)} \cdot \mathbf{y}_{\max}^{N,(i)} = 0 \tag{10}$$

and, therefore, $ACC(X_{\max}^N, Y_{\max}^N) = 0$ for all odd N and all rotation angles α. □

Observation 1 *The radii of gyration of X_{max}^N and Y_{max}^N for odd N are*

$$R_G^2(X_{max}^N) = \frac{2}{N} b^2 \sum_{i=1}^{M^-} (i)^2 \tag{11}$$

$$R_G^2(Y_{max}^N) = -b^2 \left(\frac{M^- M^-}{N}\right)^2 + \frac{1}{N} b^2 \left((M^-)^2 + 2 \sum_{i=1}^{\hat{M}^-} (i)^2 \right) \tag{12}$$

with $M^- = \frac{N-1}{2}$ and $\hat{M}^- = \frac{N-3}{2}$.

A derivation of these expressions can be found in Ref. [12]. Combining Eqs. (8), (11), and (12) yields an analytic formula for $D_{\max}(N)$ for odd N, asymptotically approaching [12]:

$$\hat{D}_{\max}(N) = \lim_{N\to\infty} D_{\max}(N) = \frac{1}{4}\sqrt{\frac{5}{3}} bN \,. \tag{13}$$

We conjecture that this asymptotic limit is valid also for even N and, since the maximum RMSD of RW chains is always larger than that of any other chain ensemble, formulate:

Conjecture 2 *$\hat{D}_{max}(N)$ is an asymptotic upper bound on the RMSD between any two linear chains.*

5 Conclusion

We combined stochastic global optimization and analytic geometry in order to conjecture an upper bound for the RMSD between linear chains of N beads with link length b after optimal roto-translational fitting. We reported pairs of putative extremal configurations of RW chains and an analytical expression for the maximum RMSD between these extremal configurations for odd N. This expression asymptotically approaches $\frac{1}{4}\sqrt{\frac{5}{3}} bN$ for large N, which is the conjectured upper bound for any two linear chains for all N. Future research will try proving this conjecture.

Acknowledgments

We thank Dr. Bojan Žagrović and Dr. Philippe Hünenberger for many insightful discussions.

References

[1] M. R. Betancourt and J. Skolnick. Universal similarity measure for comparing protein structures. *Biopolymers*, 59(5):305–309, Oct 2001.

[2] C. R. Cantor and R. Schimmel, Paul. *Biophysical chemistry Part III: The behavior of biological macromolecules*. W. H. Freeman, New York, 1980.

[3] P. J. Flory. *Statistical Mechanics of Chain Molecules*. Wiley-Interscience, New York, 1969.

[4] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[5] N. Hansen. *The CMA Evolution Strategy: A Tutorial*, Nov 2007.

[6] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallogr. A.*, 32(SEP1):922–923, 1976.

[7] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallogr. A.*, 34(SEP):827–828, 1978.

[8] G. R. Kneller. Superposition of Molecular Structures using Quaternions. *Mol. Simulat.*, 7(1):113–119, 1991.

[9] G. R. Kneller. Comment on "Using quaternions to calculate RMSD" - [J. Comp. Chem. 25, 1849 (2004)]. *J. Comp. Chem.*, 26(15):1660–1662, Nov 2005.

[10] A. D. McLachlan. Mathematical procedure for superimpsoing atomic coordinates of proteins. *Acta Crystallogr. A.*, A 28(NOV1):656–657, 1972.

[11] A. D. McLachlan. How alike are the Shapes of Two Random Chains. *Biopolymers*, 23(7):1325–1331, 1984.

[12] C. L. Müller. *Black-box Landscapes: Characterization, Optimization, Sampling, and Application to Geometric Configuration Problems*. PhD thesis, Institute of Theoretical Computer Science, Department of Computer Science, ETH Zürich, 2010.

[13] C. L. Müller, I. F. Sbalzarini, W. F. van Gunsteren, B. Zagrovic, and P. H. Huenenberger. In the eye of the beholder: Inhomogeneous distribution of high-resolution shapes within the random-walk ensemble. *J. Chem. Phys.*, 130(21), Jun 7 2009.

[14] D. C. Sullivan and I. D. Kuntz. Distributions in protein conformation space: Implications for structure prediction and entropy. *Biophys. J.*, 87(1):113–120, Jul 2004.

Angle-Restricted Steiner Arborescences for Flow Map Layout*

Kevin Buchin† Bettina Speckmann† Kevin Verbeek†

Abstract

Flow maps visualize the movement of objects between places. One or more sources are connected to several targets by arcs whose thickness corresponds to the amount of flow between a source and a target. Flow maps reduce visual clutter by merging (bundling) lines smoothly and by avoiding self-intersections.

We present algorithms that compute crossing-free flows of high visual quality. To this end we introduce a new variant of the geometric Steiner arborescence problem. The goal is to connect the targets to a source with a tree of minimal length whose arcs obey a certain restriction on the angle they form with the source. Such an *angle-restricted Steiner arborescence*, or simply *flow tree*, naturally induces a clustering on the targets and smoothly bundles arcs.

We study the properties of optimal flow trees and show that they consist of logarithmic spirals and straight lines. Computing optimal flow trees is NP-hard. Hence we consider a variant of flow trees which uses only logarithmic spirals, so called *spiral trees*. Spiral trees approximate flow trees within a factor depending on the angle restriction. Computing optimal spiral trees remains NP-hard. We present an efficient 2-approximation for spiral trees, which can be extended to avoid certain types of obstacles.

1 Introduction

Flow maps are a cartographic method for the visualization of the movement of objects between places [9]. One or more sources are connected to several targets by arcs whose thickness corresponds to the amount of flow between a source and a target. Most flow maps are drawn by hand and few automated methods exist.

Good flow maps share some common properties. They reduce visual clutter by merging (bundling) lines as smoothly and frequently as possible. Furthermore, they strive to avoid crossings between lines. Specifically, single-source flows are drawn entirely without crossings. Flow maps that depict trade often route edges along actual shipping routes. In that case a moderate distortion of the underlying geography is admissible. In contrast, flow maps that depict more abstract data, such as migration or internet traffic, do not distort the geography of the underlying map. In addition, flow maps often try to avoid covering important map features with flows to aid recognizability.

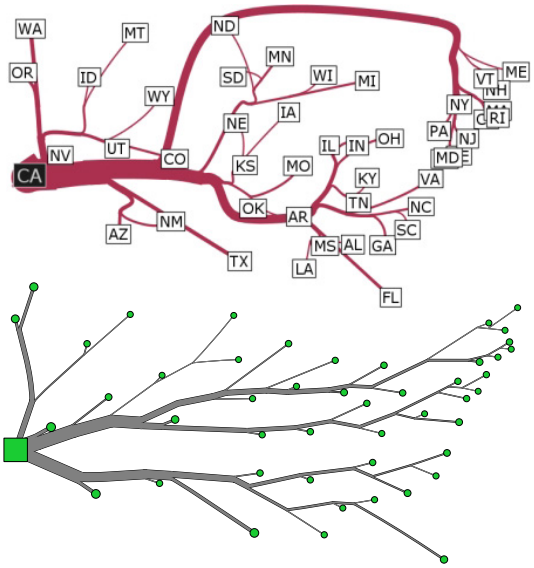


Figure 1: Two maps based on the same data set: migration from California 1995–2000. Phan *et al.* [4] (top), and the output of our algorithm (bottom).

Modeling and formal problem description. We study the problem of computing crossing-free single-source flows of high visual quality. Our input consists of a point r, the *root* or *source*, and n points $t_1, \ldots, t_n$, the *terminals* or *targets*. For every target t_i, we are also given a weight w_i, representing the amount of flow from the source to target t_i. Visually appealing flows merge quickly, but smoothly. Disregarding weights, a geometric minimal *Steiner arborescence* on our input results in the shortest possible tree, which naturally merges quickly. A Steiner arborescence for a given root and a set of terminals is a rooted directed *Steiner tree*, which contains all terminals and where all edges are directed away from the root. Without additional restrictions on the edge directions (as in the rectilinear case or in the variant proposed below), a geometric Steiner arborescence is simply a geometric Steiner tree with directed edges. Steiner arborescences have angles of $2\pi/3$ at every internal node and hence do not have the smooth appearance of hand-drawn flow maps. This motivates us to introduce a new variant of the geometric minimal Steiner arborescence problem.

*B. Speckmann and K. Verbeek are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.022.707.

†Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands, k.a.buchin@tue.nl, speckman@win.tue.nl, and k.a.b.verbeek@tue.nl

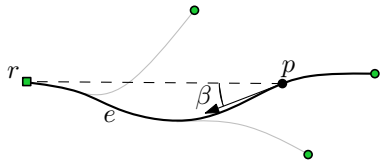


Figure 2: The angle restriction.

The goal is to connect the terminals to the root with a tree of minimal length whose arcs obey a certain restriction on the angle they form with the root.

Specifically, we use a *restricting angle* $\alpha < \pi/2$ to control the direction of the arcs of a Steiner arborescence T. Consider a point p on an arc e from a terminal to the root (see Fig. 2). Let β be the angle between the vector from p to the root r and the tangent vector of e at p. We require that $\beta \leq \alpha$ for all points p on T. We call a Steiner arborescence which obeys this angle restriction an *angle-restricted Steiner arborescence*, or simply *flow tree*. Note, that this definition is not taking weights into account. Hence an optimal flow tree is simply an angle-restricted Steiner arborescence of minimal length. Here and in the remainder of the paper it is convenient to direct flow trees from the terminals to the root. Also, to simplify descriptions, we often identify the nodes of a flow tree T with their locations in the plane.

In the context of flow maps it is important that flow trees can avoid obstacles, which model important features of the underlying geographic map. It is also undesirable that terminals become internal nodes of a flow tree. We can avoid this by covering each terminal with an obstacle. Hence our input also includes a set of m obstacles $B_1, \ldots, B_m$. We denote the total complexity of all obstacles by M. In the presence of obstacles our goal is to find the shortest flow tree T that is planar and avoids the obstacles. Finally, most flow maps use thick edges to indicate the amount of flow. To this end we consider *weighted flow trees*. The arcs of a weighted flow tree T must satisfy the following conditions [9]. If an arc e starts at a terminal t_i then the thickness of e must be proportional to w_i. If an arc e starts at an internal node u of T then its thickness must be proportional to the sum of the thicknesses of the arcs terminating in u. We require, of course, that thick arcs do not overlap obstacles.

Related work. One of the first systems for the automated creation of flow maps was developed by Tobler in the 1980s [10, 11]. His system does not use edge bundling and hence the resulting maps suffer from visual clutter. A couple of years ago Phan *et al.* [4] presented an algorithm, based on hierarchical clustering of the terminals, which creates single-source flow maps with bundled edges. This algorithm uses an iterative ad-hoc method to route edges and hence is often unable to avoid crossings.

There are many variations on the classic Steiner tree problem which use metrics that are related to their specific applications. Of particular relevance is the *rectilinear Steiner arborescence* (RSA) problem. Here we are given a root at the origin and a set of terminals $t_1, \ldots, t_n$ in the northeast quadrant of the plane. The goal is to find the shortest rooted rectilinear tree T with all edges directed away from the root, such that T contains all points $t_1, \ldots, t_n$. For any edge T from $p = (x_p, y_p)$ to $q = (x_q, y_q)$ it must hold that $x_p \leq x_q$ and $y_p \leq y_q$. If we drop the condition of rectilinearity then we arrive at the *Euclidean Steiner arborescence* (ESA) problem. In both cases it is NP-hard [7, 8] to compute a tree of minimum length. Rao *et al.* [6] give a simple 2-approximation algorithm for minimum rectilinear Steiner arborescences. Córdova and Lee [1] describe an efficient heuristic which works for terminals located anywhere in the plane. Ramnath [5] presents a more involved 2-approximation that can also deal with rectangular obstacles. Finally, Lu and Ruan [2] developed a PTAS for minimum rectilinear Steiner arborescences.

Results and organization. Section 2 gives properties of optimal flow trees. In particular, the arcs of optimal flow trees consist of (segments of) logarithmic spirals and straight lines. Flow trees naturally induce a clustering on the terminals and smoothly bundle lines. In the full paper we show that it is NP-hard to compute optimal flow trees. Hence, Section 3 introduces a variant of flow trees, so called *spiral trees*. The arcs of spiral trees consist only of logarithmic spiral segments. We prove that spiral trees approximate flow trees within a factor depending on the restricting angle α. Computing optimal spiral trees remains NP-hard. But we can establish a connection between spiral trees and rectilinear Steiner arborescences. Based on that in Section 4 we develop a 2-approximation algorithm for spiral trees which runs in $O(n \log n)$ time. In the full paper we extend our approximation algorithm to include certain types of obstacles. Finally, in Section 5 we briefly comment on weighted flow trees.

2 Optimal flow trees

Recall that our input consists of a root r, terminals $t_1, \ldots, t_n$, and a restricting angle $\alpha < \pi/2$. We can assume that the root lies at the origin. Recall further that an optimal flow tree is a geometric Steiner arborescence, whose arcs are directed from the terminals to the root and that satisfies the angle restriction.

Spiral regions. For a point p in the plane, we consider the region $\mathcal{R}_p$ of all points that are *reachable* from p with an *angle-restricted* path, that is, with a path that satisfies the angle restriction. Clearly, the root r is always in $\mathcal{R}_p$. The boundaries of $\mathcal{R}_p$ consist of curves that follow one of the two directions that form exactly an angle α with the direction towards

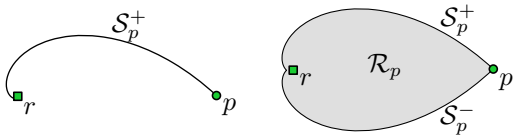


Figure 3: Spirals and spiral regions.

the root. Curves with this property are known as *logarithmic spirals* (see Fig. 3). Logarithmic spirals are self-similar; scaling a logarithmic spiral results in another logarithmic spiral. As all spirals in this paper are logarithmic, we simply refer to them as *spirals*. For $\alpha < \pi/2$ there are two spirals through a point. The *right spiral* $\mathcal{S}_p^+$ is given by the following parametric equation in polar coordinates, where $p = (R, \phi)$: $R(t) = Re^{-t}$ and $\phi(t) = \phi + \tan(\alpha)t$. The parametric equation of the *left spiral* $\mathcal{S}_p^-$ is the same with α replaced by $-\alpha$. Note that a right spiral $\mathcal{S}_p^+$ can never cross another right spiral $\mathcal{S}_q^+$ (the same holds for left spirals). The spirals $\mathcal{S}_p^+$ and $\mathcal{S}_p^-$ cross infinitely often. The reachable region $\mathcal{R}_p$ is bounded by the parts of $\mathcal{S}_p^+$ and $\mathcal{S}_p^-$ with $0 \leq t \leq \pi \cot(\alpha)$. We therefore call $\mathcal{R}_p$ the *spiral region* of p. It follows directly from the definition that for all $q \in \mathcal{R}_p$ we have that $\mathcal{R}_q \subseteq \mathcal{R}_p$.

Lemma 1 *The shortest angle-restricted path between a point p and a point $q \in \mathcal{R}_p$ consists of a straight segment followed by a spiral segment. Either segment can have length zero.*

Using Lemma 1 we establish the following properties.

Property 1 *An optimal flow tree consists of straight segments and spiral segments (see Fig. 4).*

Property 2 *Every node in an optimal flow tree, other than the root, has at most two incoming edges.*

Property 3 *Every optimal flow tree is planar.*

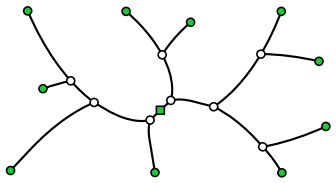


Figure 4: An optimal flow tree ($\alpha = \pi/6$).

3 Spiral trees

In this section we introduce *spiral trees* and prove that they approximate flow trees. The arcs of a spiral tree consist only of spiral segments of a given α. An optimal spiral tree is hence the shortest flow tree that uses only spiral segments. Any particular arc of a spiral tree can consist of arbitrarily many spiral segments; it can switch between following its right spiral and following its left spiral an arbitrary number of times. The length of a spiral segment can be expressed in polar coordinates. Let $p = (R_1, \phi_1)$ and $q = (R_2, \phi_2)$ be two points on a spiral, then the distance $D(p, q)$ between p and q on the spiral is

$$D(p, q) = \sec(\alpha)|R_1 - R_2|\,. \tag{1}$$

Consider now the shortest *spiral path*—using only spiral segments—between a point p and a point q reachable from p. The reachable region for p is still its spiral region $\mathcal{R}_p$, so necessarily $q \in \mathcal{R}_p$. The length of a shortest spiral path is given by Equation 1. The shortest spiral path is not unique, in particular, any sequence of spiral segments from p to q is shortest, as long as we keep moving towards the root.

Theorem 2 *The optimal spiral tree T' is a $\sec(\alpha)$-approximation of the optimal flow tree T.*

Proof. Let $\mathcal{C}_R$ be a circle of radius R with the root r as center. A lower bound for the length of T is given by $L(T) \geq \int_0^\infty |T \cap \mathcal{C}_R| dR$, where $|T \cap \mathcal{C}_R|$ counts the number of intersections between the tree T and the circle $\mathcal{C}_R$. Using Equation 1, the length of T' is $L(T') = \sec(\alpha) \int_0^\infty |T' \cap \mathcal{C}_R| dR$. Now consider the spiral tree T'' with the same nodes as T, but where all arcs between the nodes are replaced by a sequence of spiral segments (see Fig. 5). For a given circle $\mathcal{C}_R$, this operation does not change the number of intersections of the tree with $\mathcal{C}_R$, i.e. $|T \cap \mathcal{C}_R| = |T'' \cap \mathcal{C}_R|$. So we have that $L(T') \leq L(T'') \leq \sec(\alpha) L(T)$. □

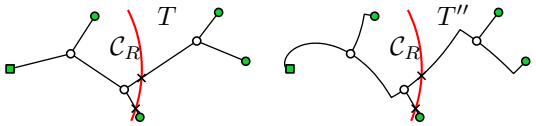


Figure 5: T and T''.

Observation 1 *An optimal spiral tree is planar and every node $u \neq r$ has at most two incoming edges.*

Relation with rectilinear Steiner arborescences. Both rectilinear Steiner arborescences and spiral trees contain directed paths, from the root to the terminals or vice versa. The edges of a rectilinear Steiner arborescence are restricted to point right or up, which is similar to the angle restriction of spiral trees. The relation between the concepts cannot be used directly, but some of the techniques developed for rectilinear Steiner arborescences can be adapted to spiral trees.

4 Computing spiral trees

In the special case that the spiral regions of the terminals are empty, i.e., if $t_i \notin \mathcal{R}_{t_j}$ for all $i \neq j$, we can use dynamic programming to compute an optimal spiral tree in $O(n^3)$ time, based on the following lemma.

Lemma 3 *If the spiral regions of all terminals are empty, then the leaf order of any planar spiral tree follows the radial order of the terminals.*

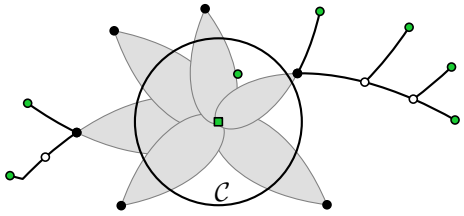


Figure 6: The wavefront $\mathcal{W}$.

We now describe an approximation algorithm that is based on a greedy algorithm for rectilinear Steiner arborescences [6]. We iteratively join nodes, possibly with Steiner nodes, until all terminals are connected in a single tree T, the *greedy spiral tree*. Initially, T is a forest. A node (or terminal) is *active* if it does not have a parent in T. In every step, we join the two active nodes for which the *join point* is farthest from r. The join point p_{uv} of two nodes u and v is the farthest point p from r such that $p \in \mathcal{R}_u \cap \mathcal{R}_v$. This point is unique if u, v, and r are not collinear.

The algorithms sweeps a circle $\mathcal{C}$, centered at r, inwards over all terminals. All active nodes that lie outside of $\mathcal{C}$ form the *wavefront* $\mathcal{W}$ (the black nodes in Fig. 6). $\mathcal{W}$ is implemented as a balanced binary search tree, where nodes are sorted according to the radial order around r. We join two active nodes u and v as soon as $\mathcal{C}$ passes over p_{uv}. For any two nodes $u, v \in \mathcal{W}$ it holds that $u \notin \mathcal{R}_v$. Hence, when $\mathcal{C}$ passes over p_{uv} and both nodes u and v are still active, then, by Lemma 3, u and v must be neighbors in $\mathcal{W}$. We process the following events.

Terminal. When $\mathcal{C}$ reaches a terminal t, we add t to $\mathcal{W}$. We need to check if there exists a neighbor v of t in $\mathcal{W}$ such that $t \in \mathcal{R}_v$. If such a node v exists, then we remove v from $\mathcal{W}$ and connect v to t. Finally we compute new join point events for t and its neighbors in $\mathcal{W}$.

Join point. When $\mathcal{C}$ reaches a join point p_{uv} (and u and v are still active), we connect u and v to p_{uv}. Next, we remove u and v from $\mathcal{W}$ and we add p_{uv} to $\mathcal{W}$ as a Steiner node. Finally we compute new join point events for p_{uv} and its neighbors in $\mathcal{W}$.

We store the events in a priority queue $\mathcal{Q}$, ordered by decreasing distance to r. It is easy to verify that the total number of events is $O(n)$, and that each event can be handled in $O(\log n)$ time, so the total running time is $O(n \log n)$. The greedy spiral tree is planar by construction. We can prove the following results.

Lemma 4 *Let $\mathcal{C}$ be any circle centered at r and let T and T' be the optimal spiral tree and the greedy spiral tree, respectively. Then $|\mathcal{C} \cap T'| \leq 2|\mathcal{C} \cap T|$.*

Theorem 5 *The greedy spiral tree is a 2-approximation of the optimal spiral tree and can be computed in $O(n \log n)$ time.*

5 Weighted flow trees

When creating flow maps we need to consider weighted flow trees, that is, flow trees with thick arcs. To facilitate easy comparisons between flows these arcs should be drawn as thickly as possible. However, increasing the thickness can increase the length of the optimal flow tree substantially. This tradeoff between arc thickness and tree length makes it very difficult to produce theoretically interesting results regarding weighted flow trees. Hence we implemented a heuristic approach based on our approximation algorithm for (thin) spiral trees. We thicken the edges of the greedy spiral tree, pushing arcs away from terminals and obstacles, and apply local changes to the tree topology to facilitate thicker flows. The resulting flow maps are still guaranteed to be crossing free. First results look very promising (see Fig. 1); our maps are less cluttered than those produced by other automated methods, and we can observe that spiral trees naturally cluster terminals well.

References

[1] J. Córdova and Y. Lee. A Heuristic Algorithm for the Rectilinear Steiner Arborescence Problem. Technical Report, Engineering Optimization, 1994.

[2] B. Lu and L. Ruan. Polynomial Time Approximation Scheme for the Rectilinear Steiner Arborescence Problem. *Journal of Combinatorial Optimization*, 4(3):357–363, 2000.

[3] J. Mitchell. L_1 shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8(1):55–88, 1992.

[4] D. Phan, L. Xiao, R. Yeh, P. Hanrahan and T. Winograd. Flow map layout. In *Proc. IEEE Symposium on Information Visualization*, pp. 219–224, 2005.

[5] S. Ramnath. New approximations for the rectilinear Steiner arborescence problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(7):859–869, 2003.

[6] S.K. Rao, P. Sadayappan, F.K. Hwang and P.W. Shor. The rectilinear Steiner arborescence problem. *Algorithmica*, 7(1):277–288, 1992.

[7] W. Shi and C. Su. The Rectilinear Steiner Arborescence Problem Is NP-Complete. In *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms*, pp. 780–787, 2000.

[8] W. Shi and C. Su. The Rectilinear Steiner Arborescence Problem Is NP-Complete. *SIAM Journal on Computing*, 35(3):729–740, 2005.

[9] T.A. Slocum, R.B. McMaster, F.C. Kessler and H.H. Howard. Thematic Cartography and Geovisualization. Pearson, New Jersey, 2010.

[10] *CSISS - Spatial Tools: Tobler's Flow Mapper.* http://www.csiss.org/clearinghouse/FlowMapper/.

[11] W. Tobler. Experiments in Migration Mapping by Computer. *The American Cartographer*, 14(2):155–163, 1987.

Distance between Folded Objects

Chen Gu* Leonidas Guibas†

Abstract

Geometric folding problems have recently attracted much attention in both mathematics and theoretical computer science. In this paper, we study the following basic problem: given a set of folded conformations of a unit-length rope in 1D, how do we decide which are more similar or less similar to each other? We first define a distance function between flat folded states that incorporates both the geometry and the overlap order. Then we do some computational experiments clustering random folded ropes with this distance metric. Finally, we generalize our results for 1D folded ropes to flat folded papers (origami) in 2D.

1 Introduction

We start by introducing the folding model and relevant definitions from [3]. The notion of folding ropes is very intuitive. Mathematically, we imagine the rope as a one-dimensional line segment that has zero thickness. A folding motion is a continuous motion of the rope from one configuration to another that does not cause the rope to stretch, tear, or self-penetrate. A snapshot of this motion at a particular time is called a folded state.

Since ropes are not rigid, all folded conformations (final folded states) are flat foldings. A flat folding has the property that it lies in the same space as the unfolded line segment. However, there can be multiple layers of the segment at a point, so the folding really occupies a finite number of infinitesimally close copies of the one-dimensional space. Formally, a flat folded state can be specified by a function mapping all points to their folded positions, together with a partial order that specifies their overlap order. For every pair of distinct noncrease points that are mapped into the same position, this partial order should specify which one lies above the other.

In this paper, we consider a new folding problem: determine which folded conformations are more similar or less similar to each other. This problem is very fundamental in the area of geometric folding and fairly natural from a practical point of view. Traditionally, there are many distance functions that can measure the similarity between different geometric shapes [5]. However, the difficulty of comparing folded conformations is that we cannot extract from the geometry the relative stacking order of portions of the line segment that come into contact as multiple overlapping layers. This lack of information makes it impossible to identify different folded states. Thus, we need to define a distance function that incorporates both the geometry and the ordering.

There have been several related studies on folding one-dimensional strings in recent years. For example, Arkin et al. solved the 1D flat-foldability problem that characterize which crease patterns and mountain-valley assignments have flat folded states [1], and Cardinal et al. investigated the folding complexity about how to quickly fold a paper strip to obtain a desired mountain-valley pattern of equidistant creases [2]. In particular, we will use the results of the 1D flat-foldability problem to design a sampling algorithm in our experiment of clustering random folded ropes with this distance metric.

2 Distance between folded ropes in 1D

In this section, we define a distance function to measure the similarity between different folded conformations of a unit-length rope, and develop an efficient algorithm for the distance computation.

2.1 Distance function

In order to compare flat foldings, we discretize the rope into $2n$ points, uniformly distributed on both sides of the rope (see Figure 1). We use a plus sign superscript to denote the points on the top side, and a minus sign superscript to denote the points on the bottom side. Now, consider the two conformations shown in Figure 2: suppose we fold the rope at position 3, but in two different directions. In 2(a), point 4 is connected to point 2^-, but not point 2^+; In 2(b), point 4 is connected to point 2^+, but not point 2^-. Therefore, although the geometric positions of the two discretized point sets are identical, we could distinguish these two conformations from their different topological connections.

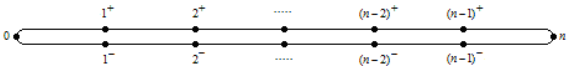


Figure 1: Discretization

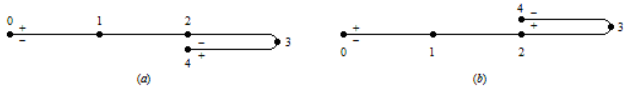


Figure 2: Mountain/Valley fold at position 3

*Institute for Computational and Mathematical Engineering, Stanford University, guc@stanford.edu

†Department of Computer Science, Stanford University, guibas@cs.stanford.edu

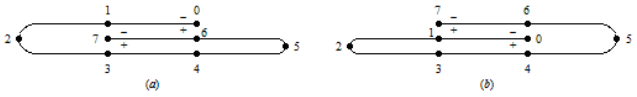


Figure 3: Two conformations with different overlap orders

We define the distance function using the distance root mean squared error (dRMS). dRMS is a metric of distance between two point sets with known correspondence, which is computed by comparing all internal pairwise distances of the two point sets:

$$dRMS^2(P,Q) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(||p_i - p_j|| - ||q_i - q_j||\right)^2$$

For simplicity, we assume that n is large enough so that we only fold the rope at integer sample points $\{1, 2, ..., n-2, n-1\}$ (or we can approximate all folding operations at these positions). Given two folded conformations, we compute their distance using dRMS, where $p_i, p_j, q_i, q_j \in \{0, (1/2)^{\pm}, 1^{\pm}, (3/2)^{\pm}, 2^{\pm}, ..., (n-1)^{\pm}, (n-1/2)^{\pm}, n\}$. (We define $(i+1/2)^{\pm}$ as the midpoint of $i^{\pm}$ and $(i+1)^{\pm}$.) The internal distance $||p_i - p_j||$ is defined as the geodesic distance $d(p_i, p_j)$ from p_i to p_j along the rope. For two points that come in contact in the folded conformation, we consider them to be connected, so that the distance between two contacted points is 0. For example, consider the two conformations shown in Figure 3: we denote $h = 1/n$ as the length between two consecutive integer sample points along the rope. In 3(a), since 0 and 6^- are connected, we have $d(0, 6^-) = 0$; However, in 3(b), $d(0, 6^-) = 2h$ since the shortest path from 0 to 6^- along the rope is $0 \to 1^-(7) \to 6^-$. Intuitively, the two conformations in Figure 3 have very similar geometric shapes, however, it might follow a long trajectory if we want to deform from one shape to another since they have different overlap orders. Using this distance function, we can check that the distance increases monotonically as we increase the overlap of the top two layers.

2.2 Algorithm

To compute internal pairwise distances, we build a graph of $4n$ nodes corresponding to all sample points $\{0, (1/2)^{\pm}, 1^{\pm}, (3/2)^{\pm}, 2^{\pm}, ..., (n-1)^{\pm}, (n-1/2)^{\pm}, n\}$. For any two consecutive sample points along the rope, we connect them by an edge with weight $h/2$. Thus, the graph of an unfolded rope is simply a cycle of $4n$ nodes. For two sample points that come in contact by a folding operation, we add an edge between them with weight 0. Then, the internal distance between any two sample points along the rope is the same as the shortest path distance between their corresponding nodes in this graph.

This graph has a property that it is very sparse. In fact, each node has exactly 2 incident edges of weight $h/2$ and at most 2 incident edges of weight 0: for any sample point, it belongs to at most 2 unit segments on the rope (a unit segment has length h with two endpoints at integer positions). For each segment, there can only be one new segment lying immediately above/below (depending on its side) it in the folded conformation. Therefore, each sample point is connected to at most 2 other sample points with edges of weight 0 from these new segments. As a result, the degree of each node in this graph is at most 4, which means the total number of edges $m = O(n)$.

For shortest paths computation, since there are only two types of different weights in the graph (0 and $h/2$), the shortest paths distances between all pairs of nodes are multiples of $h/2$ between 0 and 1. From this observation, when we implement the Dijkstra's algorithm from a single source, we can simply build an array of $2n+2$ cells, with cell i pointing to a linked list of nodes that have current upper bound on distance equal to $ih/2$. Initially, the source is linked to cell 0, while all other nodes are linked to cell $2n+1$ (which serves as infinity). Each time we update the upper bound on distance of some node by relaxing an edge, this node moves left to a linked list of nodes that have smaller distance. Using this data structure, the total amount of time is bounded by $O(m)$ for time we spent on relaxing edges plus $O(n)$ time we spent moving right in the array, looking for the next non-empty cell. Thus, the total running time of Dijkstra's shortest path algorithm is $O(m)$ in this case [4], instead of $O(m + n \log n)$ using Fibonacci heaps. Finally, since $m = O(n)$, we can implement a single source Dijkstra's algorithm in $O(n)$ time, and compute all internal pairwise distances in $O(n^2)$ time.

2.3 Symmetries

There are eight types of symmetries for folding ropes (see Figure 4). The two conformations in each column are identical, which only differ by a rotation of 180 degrees. Type (a) and type (b) differ by a horizontal reflection: if we can swap the top/bottom sides of the rope, these two types become identical. Type (a) and type (c) differ by a vertical reflection: if we can number the sample points in a reverse order from right to left, these two types become identical. Finally, type (d) contains both a horizontal and a vertical reflection from type (a).

When we do not distinguish the two ends of the rope, the conformations in type (a) and type (d) are considered as the same state. In this case, we can define the distance between two conformations C_1 and C_2 as $\min\{d(C_1, C_2), d(C_1, C_2^{rot})\}$, where C_2^{rot} corresponds to a type (d) rotation on C_2. Then the new distance function would be invariant under rotations. Furthermore, if we also want to include the symmetry of reflections, we can define the distance between two conformations as the minimum of four distances under all possible rotations/reflections.

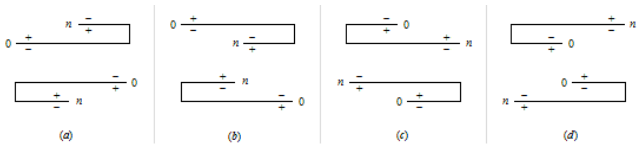


Figure 4: Symmetries

3 Experiment: clustering folded ropes

In this section, we test the performance of the distance function we defined in Section 2. We first design a sampling algorithm to generate random folded conformations of a rope, and then use this distance metric to cluster similar conformations together.

3.1 Generating random folded conformations

In our experiment, we generate random folded conformations of a rope from 1D crease patterns. The characterization of flat-foldable crease patterns has been studied extensively in origami science [3]. In 1D, a crease pattern is a set of prescribed crease points labeled with mountain or valley directions on a line segment. In the model of simple foldings, a flat folding is made by a sequence of simple folds, each of which folds one or more layers of the line segment. In [1], Arkin et al. showed that a 1D mountain-valley pattern is flat foldable if and only if it can be folded by a sequence of crimps and end folds.

Formally, let c_1,...,c_n denote the creases on a line segment, oriented from left to right. In addition, let c_0 denote the left end and c_{n+1} denote the right end. First, a pair (c_i, c_{i+1}) of consecutive creases is crimpable if c_i and c_{i+1} have opposite directions and $|c_{i-1} - c_i| \geq |c_i - c_{i+1}| \leq |c_{i+1} - c_{i+2}|$. Crimping such a pair corresponds to fold c_i and then fold c_{i+1}, using one-layer simple folds. Second, c_0 is a foldable end if $|c_0 - c_1| \leq |c_1 - c_2|$, and c_{n+1} is a foldable end if $|c_{n-1} - c_n| \geq |c_n - c_{n+1}|$. Folding such an end corresponds to perform a one-layer simple fold at its nearest crease. In the 1D flat-foldability test, we only need to search for one of these two local operations to perform. Moreover, the sequence can be found in any greedy way.

Using this result, we can generate random folded conformations as follow: we first randomly generate a crease pattern by sampling folding positions and mountain/valley directions uniformly on the rope, and then run the flat-foldability test to check whether this crease pattern can be folded flat. We repeatedly search for a local minimum segment and test whether we can fold it using crimp or end fold, until we reach the final flat folding or we know it cannot be folded flat. When the flat-foldability test fails, we simply generate a new crease pattern and test it again.

Notice that some crease patterns may have multiple flat folded states (for example, see Figure 3). The reason is that their sequences of crimps and end folds are different so that they have different overlap orders in the fold states. From this observation, we randomly select an allowable crimp or end fold operation in each step during the flat-foldability test. Using this sampling technique, we would be able to generate all valid crease patterns and folded conformations. Finally, if we want to avoid duplicated folded conformations, we can use our distance function to check whether the distance between the new folded conformation and any previous conformation is equal to 0.

3.2 Clustering result

Using the dRMS metric, each conformation maps to a point in dimension N^2 whose coordinates are its pairwise internal distances. Thus, we can apply standard clustering algorithms to group similar conformations together. Notice that when we allow rotations or reflections in the distance computation, each conformation may correspond to multiple points in the dRMS space. In this case, we can use clustering algorithms which do not require coordinate representations.

In our implementation, we set $n = 20$ and generate 40 random conformations by at most 5 folds. We do not distinguish the two ends of the rope so that two conformations are considered as the same state if they only differ by a type (d) rotation (see Figure 4). For clustering, we select cluster centers using the farthest-first traversal algorithm, which gives a 2-approximation solution for the k-center problem by repeatedly picking a new center with maximum distance to its nearest center in the last iteration.

Figure 5 shows the experiment result with 10 clusters. We see that it works fairly well for clustering similar folded conformations. Generally, our distance function will ignore small crimps and end folds, and compare the folding structures of long segments. If two conformations have the same overlap order with similar lengths on corresponding long segments, or different overlap orders but the overlap length is small, then their distance should be small; Otherwise, they are more likely to be in different clusters.

4 Extension to flat folded papers in 2D

In this section, we generalize our results for 1D simple folding to orthogonal folding in 2D. In the orthogonal folding model, we only fold along horizontal or vertical directions on a square (or rectangular) piece of paper, where horizontal and vertical are defined by the sides of the paper. In origami science, this model is also called the map folding.

Given a square paper in 2D, we discretize it by using a uniform grid on both sides of the paper (see Figure 6(a)). Similarly, we can define the distance between two flat folded papers using the dRMS distance function (under rotations/reflections if necessary). In particular, we define the internal distance between two sample points by their geodesic Manhattan distance along the paper. For example, the distance between p^+ and q^+ is $2h$ ($h = 1/n$), however, the distance between p^+ and q^- is $4h$ since we need to go across the boundary of the paper. Again, for two points that come in contact in the folded conformation, we consider them to be connected, so that their internal distance along the paper is 0.

To compute internal pairwise distances, we build a graph consists of all sample points at $(ih/2, jh/2)^{\pm}$ ($n + 1$ integer positions and n midpoints along each axis) on both sides of the paper. Each sample point is connected to its 4 neighbors in the grid by edges of weight $h/2$, and at most 4 contacted points by edges

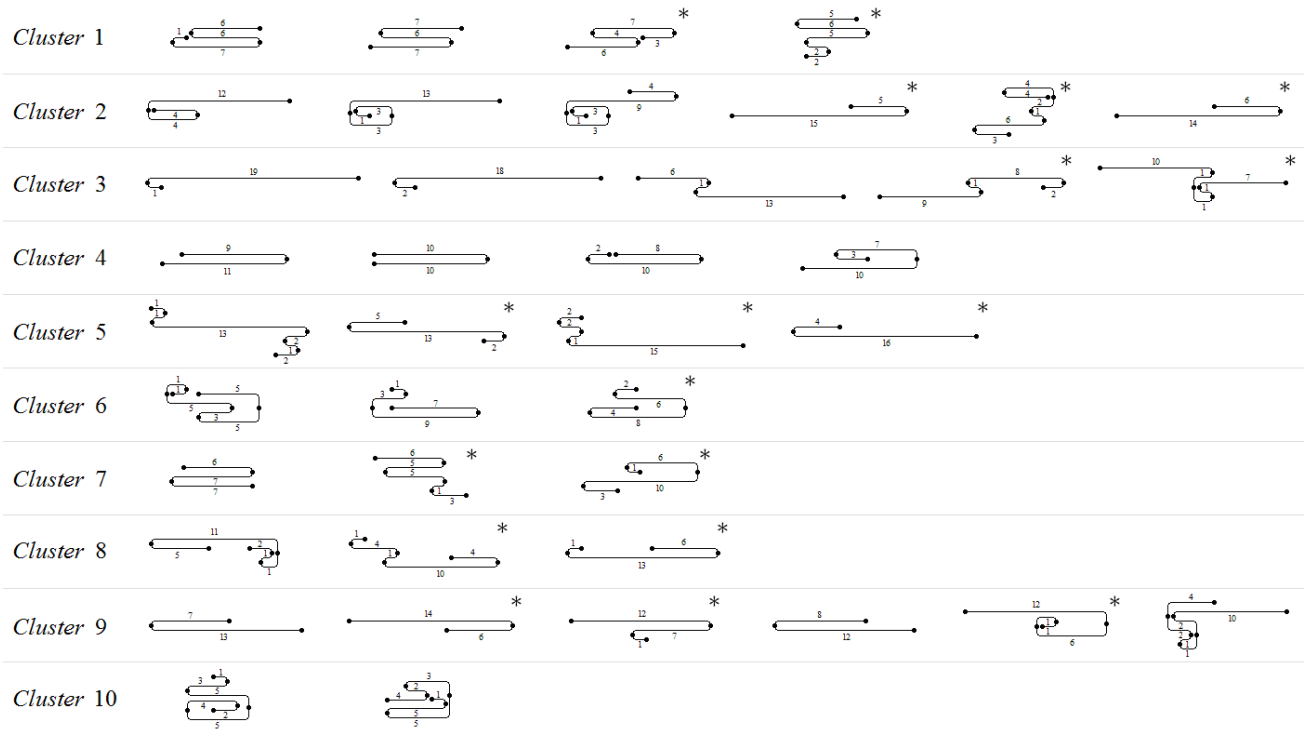


Figure 5: Experiment result: in each cluster, the first conformation represents the cluster center computed by the farthest-first traversal algorithm, and all other conformations are sorted in ascending order by their distances to this cluster center. (Conformations marked with a '*' are rotated by 180 degrees when compared with the cluster centers.)

of weight 0. (we check the 4 unit squares around a sample point to see if there are any unit squares lying immediately above/below (depending on their sides) them in the folded conformation, see the shaded areas around p in Figure 6(a)). Therefore, the degree of each node in this graph is $O(1)$, and the total number of edges $m = O(n^2)$. Using a similar data structure from the 1D algorithm, we can implement a single source Dijkstra's algorithm in $O(n^2)$ time, and compute all internal pairwise distances in $O(n^4)$ time.

In addition, we can also allow folding operations to be axis-parallel plus at a 45-degrees angle (see Figure 6(b)). In this case, we can still use the geodesic Manhattan distance for internal distances computation. The only difference is that we may need to check at most 8 unit triangles around a sample point to build edges for connected points in the folded conformation.

When we allow folding operations at arbitrary directions, the geodesic Manhattan distance does not work because the grid may not be aligned after a simple fold. In this case, we may define the internal distance between two sample points as their geodesic Euclidean distance along the paper. However, since the shortest path connecting two points along the paper might be at any direction, we need to build a complete graph of all sample points, and the computation for their geodesic Euclidean distance might be very complex in a complicated folded conformation. The case of arbitrary folding in 2D might be a topic for our future research.

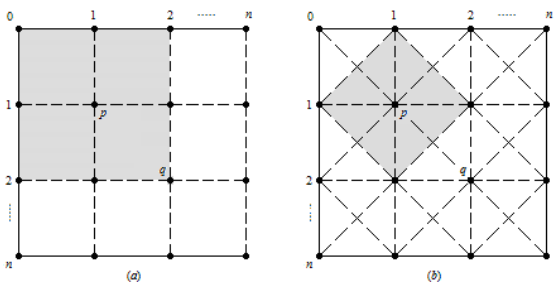


Figure 6: 2D folding (a) orthogonal (b) diagonal

Acknowledgments

This research was supported by NSF grant IIS-0914833.

References

[1] E. Arkin, M. Bender, E. Demaine, M. Demaine, J. Mitchell, S. Sethia, and S. Skiena. *When can you fold a map?* Computational Geometry: Theory and Applications, 29(1): 23-46, 2004.

[2] J. Cardinal, E. Demaine, M. Demaine, S. Imahori, S. Langerman, and R. Uehara. *Algorithmic folding complexity.* In Proceedings of the 20th Annual International Symposium on Algorithms and Computation, Lecture notes in Computer Science, volume 5878, pages 452-461, 2009.

[3] E. Demaine and J. O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra.* Cambridge University Press, 2007.

[4] H. Gabow. *Scaling algorithms for network problems.* Journal of Computer and System Sciences, 31:148-168, 1985.

[5] L. Kavraki. *Molecular distance measures.* Connexions, http://cnx.org/content/m11608/1.23/, 2007.

Improving shortest paths in the Delaunay triangulation *

Manuel Abellanas† Mercè Claverol‡ Gregorio Hernández† Ferran Hurtado§ Vera Sacristán§
Maria Saumell§ Rodrigo I. Silveira§

Abstract

We study a problem about shortest paths in Delaunay triangulations. Given two nodes s, t in the Delaunay triangulation of a point set P, we look for a new point p that can be added, such that the shortest path from s to t, in the Delaunay triangulation of $P \cup \{p\}$, improves as much as possible. We study several properties of the problem, and give efficient algorithms to find such point when the graph-distance used is Euclidean and for the link-distance. Several other variations of the problem are also discussed.

1 Introduction

There are many applications involving communication networks where the underlying physical network topology is not known, too expensive to compute, or there are reasons to prefer to use a logical network instead. An example of an area where this occurs is ad-hoc networks, where nodes can communicate with each other when their distance is below some threshold. Even though the routing is done locally, to avoid broadcasting to all neighbors every time a packet needs to be sent, some logical network topology and routing algorithm must be used.

Similar situations arise in application-layer routing, where sometimes a logical network is used on top of the actual, physical network (see for example [5]). This logical network is assumed to have some overlay topology, whose choice can have an important impact on the overall performance of the network.

The Delaunay triangulation is often used to model the overlay topology [3, 5] due to several advantages: it provides locality, scales well, and in general avoids high-degree vertices, which can create serious bottlenecks. In addition, several widely-used *localized* routing protocols guarantee to deliver the packets when the underlying network topology is the Delaunay triangulation [2]. Furthermore, there are localized routing protocols based on the Delaunay triangulation where the total distance traveled by any packet is never more than a small constant factor times the network distance between source and destination (e.g. [2]). Since the Delaunay triangulation is known to be a spanner [4], in the case of geometric networks this guarantees that all packets travel at most a constant times the minimum travel time.

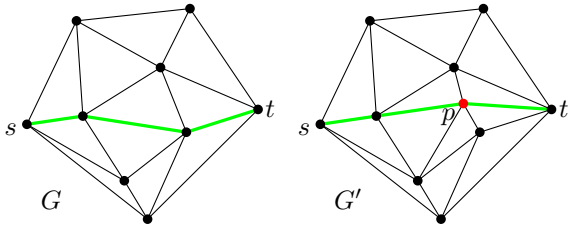


Figure 1: Shortest path between s and t before (left) and after (right) adding p, resulting in a shorter path.

In this paper we consider the problem of improving a geometric network, with a Delaunay triangulation topology, by augmenting it with additional nodes. In particular, we aim at improving the shortest path on the Delaunay network between two given nodes s and t. Adding new nodes to a Delaunay network produces changes in the network topology that can result in equal, shorter, or longer shortest paths between s and t (see Figure 1).

We restrict ourselves to the scenario where at most one node can be added to the network, which can be placed anywhere on the plane. The goal is to find a location for the new node that improves the shortest path between s and t as much as possible. We are not aware of any previous work on this problem.

Notation The input to the problem is a set of n points $P = \{p_1, \cdots, p_n\}$, and two points $s, t \in P$. The points represent the locations of the network nodes.

We will use G to denote the Delaunay graph of P: G has the points in P as vertices, and an edge between two vertices (p_i, p_j) if and only if there is a circle through p_i, p_j that does not contain any point

*This research was initiated during the 6th Iberian Workshop on Computational Geometry, held in Aveiro, Portugal. M.A. and G.H. were partially supported by projects MTM2008-05043 and HP2008-0060. M.C., F.H., V.S. and M.S. were partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040. R.I.S. was supported by the Netherlands Organisation for Scientific Research (NWO).

†Facultad de Informática, Universidad Politécnica de Madrid, {mabellanas, gregorio}@fi.upm.es.

‡Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, merce@ma4.upc.edu

§Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, {ferran.hurtado, vera.sacristan, maria.saumell, rodrigo.silveira}@upc.edu

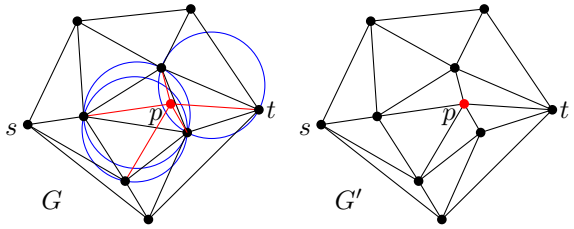


Figure 2: Adding a new point p to the Delaunay triangulation.

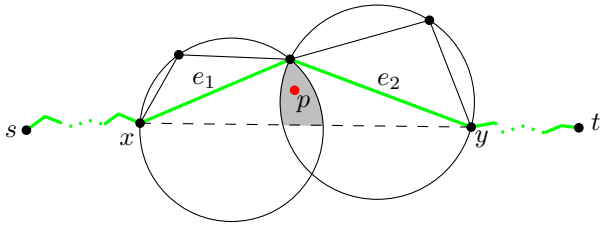


Figure 3: Any point inserted in the shaded region, like p, improves $SP(s,t)$, shown in green.

from P in its interior. We assume the points in P are in general position: no three points are collinear and no four points are cocircular. Thus G represents the Delaunay triangulation of P. Moreover, we also assume that the edge $(s,t) \notin G$, otherwise the distance between s and t in G would be optimal. Note that we use G to refer to both the graph and the triangulation.

The shortest path on G between s and t will be denoted by $SP_G(s,t)$. The length of such path, defined as the sum of the Euclidean lengths of its edges, will be denoted by $|SP_G(s,t)|$, although we will omit G if possible. The straight line segment between two points x and y will be denoted by $\overline{xy}$, and its Euclidean length by $|\overline{xy}|$, whereas the Euclidean length of an edge (x,y) will be denoted by $|(x,y)|$.

Finally, we will use G'_p to denote the Delaunay graph of $P \cup \{p\}$, for some $p \notin P$ (we will omit p when clear from the context).

2 Properties and observations

We begin the paper by analyzing some geometric properties of the problem.

When a new point p is inserted in P, some edges of the Delaunay triangulation might disappear and new edges, all incident to p, appear (see Figure 2). The edges of G that are affected by the insertion of p belong to Delaunay triangles whose circumcircles contain p. In particular, all triangles in G whose circumcircle contains p get new edges in G', connecting their vertices to p. If p is outside the convex hull of P, some additional edges might appear.

A first question that one may ask is whether it is always possible to improve a shortest path by adding one point to G. There are situations in which it is easy to obtain *some* improvement:

Lemma 1 *Let e_1 and e_2 be two consecutive edges of $SP_G(s,t)$. Let C_1 and C_2 be two Delaunay circles through the extremes of e_1 and e_2, respectively. If C_1 and C_2 are not tangent and $C_1 \cap C_2$ is on the side in which the edges form the smallest angle, then the length of $SP_G(s,t)$ can always be reduced by inserting one point.*

Even though we omit the proof of the previous lemma, the basic idea can be seen in Figure 3. If we insert p in the shaded region, then (x,p) and (p,y) will be Delaunay edges in G', shortening the part of $SP(s,t)$ between x and y.

However, some shortest paths cannot be improved at all by adding a single point:

Lemma 2 *It is sometimes impossible to improve $SP_G(s,t)$ by inserting only one point.*

Proof. An example is shown in Figure 4. Notice that e_1, e_2, C_1, and C_2 do not satisfy the hypothesis of Lemma 1. Moreover, $\forall x \in P \setminus \{s,t\}$, $|xs| + |xt| \geq |e_1| + |e_2|$, so a shorter path between s and t must be of the form $\{s,p,t\}$. More precisely, p must be inserted so that $|sp| + |pt| < |e_1| + |e_2|$, $(s,p) \in G'_p$, and $(t,p) \in G'_p$. It is easy to verify that these conditions cannot be simultaneously satisfied. □

The example in Figure 4 can be extended to a path with a linear number of vertices, and even to one where $SP(s,t)$ zigzags.

On the other hand, two points always suffice to improve the shortest path between s and t.

Lemma 3 *It is always possible to reduce the length of $SP_G(s,t)$ by inserting two points, if $SP_G(s,t)$ is not optimal.*

Proof. Let $e_1 = (a,b)$, $e_2 = (b,c)$, C_1 and C_2 be two consecutive edges of $SP(s,t)$, and two Delaunay cir-

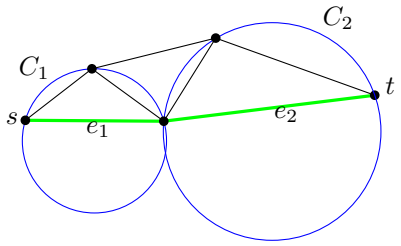


Figure 4: Example where $SP(s,t)$ cannot be improved by adding one point.

cles circumscribing them. If Lemma 1 cannot be applied, it is easy to see that we can place two points p_1 and p_2 close enough to b so that $\{a, p_1, p_2, c\}$ is a path in the new triangulation, shorter than $\{a, b, c\}$. □

3 Finding a point that gives the maximum improvement

In this section we present an algorithm that computes a point p such that $|SP_{G'_p}(s,t)|$ is minimum. The correctness of the algorithm is based on the following lemmas. First we prove that we only need to look at $O(I)$ possible candidate points, where I is the number of pairs of Delaunay circles that intersect—$I \in \Theta(n^2)$ in the worst case. Then we show that the shortest paths from s and t need to be computed only once.

Lemma 4 *Let p be an optimal point, and let x, y be the points in G such that $SP_{G'_p}(s,t)$ includes (x,p) and (p,y) as consecutive edges. Then p lies on the segment $\overline{xy}$, or on the intersection of a circumcircle through x, and a circumcircle through y.*

Proof. Suppose that p does not lie on the straight line segment $\overline{xy}$. Assume w.l.o.g. that $\overline{xy}$ is horizontal, and p is above $\overline{xy}$. Since p is optimal but moving p down reduces $|xp| + |py|$, any movement toward $\overline{xy}$ must result in crossing a circle C that causes a flip removing (x,p), (p,y) or some other edge in $SP_{G'}(s,t)$. This can be because p either enters or leaves C.

If p enters a circle, an edge e connecting two neighbors of p disappears, and is replaced by an edge e' incident to p. Such a change cannot affect the combinatorial structure of the current shortest path $SP_{G'}(s,t)$.

Hence we are only interested in the event of leaving a circle C. In this case an edge e, until now incident to p, disappears, and is replaced by another edge e' (see Figure 5). This can be a problem only if $e = (x,p)$ or $e = (p,y)$. Assume that $e = (x,p)$. Notice that x lies on C. Even though moving p down would make it cross C, the length of $SP_{G'}(s,t)$ could also be reduced by moving p on the circle C toward $\overline{xy}$. If such a movement cannot be done without affecting the combinatorial structure of $SP_{G'}(s,t)$, we conclude that there is a second circle C' through p and y. □

Lemma 5 *Let p be a point, and let $x \in P$ such that $(x,p) \in G'_p$. If $SP_{G'_p}(s,p)$ includes (x,p), then $|SP_{G'_p}(s,p)| = |SP_G(s,x)| + |xp|$. Otherwise, $|SP_{G'_p}(s,p)| \leq |SP_G(s,x)| + |xp|$.*

Algorithm The previous lemmas imply that to find an optimal point p it is enough to analyze each pair of Delaunay circles that intersect.

We first precompute the shortest path trees from s and from t. Then we use an output-sensitive algorithm to compute all pairs of circles that intersect.

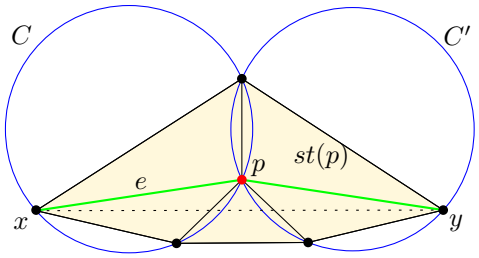


Figure 5: Situation in proof of Lemma 4.

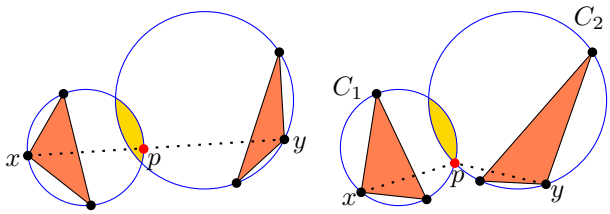


Figure 6: Two optimal ways to connect x and y: by a point on $\overline{xy}$ (left), and by a point on the boundary of $C_1 \cap C_2$ (right).

For each pair of intersecting circles (C_1, C_2), we proceed as follows. Each circle corresponds to a Delaunay triangle from G. Let the two triangles be t_1, t_2. For each pair of vertices $x \in t_1$ and $y \in t_2$, we first check if $\overline{xy}$ intersects $C_1 \cap C_2$. If it does, we take p as any point on $(\overline{xy} \cap C_1 \cap C_2)$. Otherwise, we check the two points where C_1 and C_2 intersect, and use the one that gives the shortest path from x to y. See Figure 6.

If the length of $SP(s,t)$ improves by using p, we update this information. In the end we output the point that gave the shortest path, if it improves over $SP_G(s,t)$, or report that no point can improve it.

The running time of the algorithm is dominated by the time needed to find all pairs of circles that intersect. Therefore by using an algorithm like Balaban's [1] we obtain an $O(n \log n + I)$ running time, with $O(n)$ space.

Theorem 6 *Given the Delaunay triangulation of n points, and two vertices s and t, a point whose insertion gives the maximum improvement in $SP(s,t)$ can be found in $O(n \log n + I)$ time, where I is the number of pairs of Delaunay circles that intersect.*

3.1 Related problems

Based on the previous lemmas, some other related problems can also be solved more efficiently.

Finding an optimal point for every t If only s is fixed, one may want to compute for each $t \in P$ a

point p_t whose insertion gives the optimal improvement in $SP(s,t)$. This can be done efficiently in two steps. First we augment G by adding some edges as follows: we add an edge (x,y) of weight w if there exists a circumcircle through x intersecting a circumcircle through y, such that the best point p in the intersection satisfies $|xp|+|py|=w$. These new edges can be found in $O(n\log n+I)$ time, and the resulting graph H has $O(n+I)$ edges and n vertices. In the second step, we use Dijkstra's algorithm implemented with Fibonacci heaps to compute the single-source shortest paths from s in H, modified to ensure that no path uses more than one of the new edges. This yields a running time of $O(n\log n+I)$. This approach can also be used to compute, for each s and each t, the point p_{st} that gives the optimal improvement for s and t, in $O(n^2\log n+nI)$ time.

Other network graphs If instead of the Delaunay graph a different proximity graph is used (e.g. Gabriel or nearest neighbor graph), one can apply the general approach of partitioning the plane into regions such that inserting a point anywhere inside the region produces the same topological change to the structure. Then the exact optimal location within a region can be computed. Several proximity graphs related to the Delaunay graph can use such technique, including the minimum spanning tree (the corresponding subdivision has been studied in [6]).

4 Finding a point that gives the maximum improvement, using the link-distance

Next we consider the variant of the problem where the metric used to measure distances on G is the link-distance: the length of a path is defined by its number of edges. This metric is also interesting in networking applications, since it measures the number of hops.

We use $d_l(s,t)$ to denote the link-distance between s and t. As before, we are interested in adding one new point to G such that the link-distance between s and t is minimized as much as possible.

Data structure We use a data structure D that allows to answer the following type of query in $O(\log^2 n)$ time: given a circle C, find a Delaunay circle C^* that (i) intersects C, and (ii) has minimum link-distance to t. The link-distance from a circumcircle C—corresponding to a triangle $\triangle(a,b,c)$—to t is defined as $\min\{d_l(a,t),d_l(b,t),d_l(c,t)\}$. The data structure consists of a balanced binary tree where each node represents the union of a subset of the circumcircles. The unions are represented by additively-weighted Voronoi diagrams of the circle centers. It takes $O(n\log n)$ space and can be built in $O(n\log^2 n)$ time. We omit the details due to the space limit.

Algorithm The algorithm proceeds as follows. First the shortest paths from s and t to each other node are precomputed. Then we go over all points in P. For each possible point x we query the data structure D with each circumcircle that goes through x. We simply keep track of the lowest value returned by each query, for each point. After doing this for all points in P, we return a point in the intersection associated with the circles that gave the minimum distance. The correctness of the algorithm follows from the previous lemmas, which also hold for the link-distance. The total running time of the algorithm is $O(n\log^2 n)$.

Theorem 7 *Given the Delaunay triangulation of n points, and two vertices s and t, a point whose insertion gives the maximum improvement in $SP(s,t)$, under the link-distance, can be found in $O(n\log^2 n)$ time.*

5 Discussion

We studied the problem of adding a point to a Delaunay triangulation, such that it improves a shortest path as much as possible. As already mentioned, the methods and observations used can be adapted to solve other related problems, like when other proximity graphs are used. It is also possible to solve efficiently the somewhat dual problem of finding a node whose removal gives the best improvement to the shortest path between s and t. We omit details due to space constraints.

Several improvements to our algorithms are possible. A particularly intriguing question is whether the decision problem (Is there a point that improves $SP(s,t)$?) can be solved faster than the optimization version.

References

[1] I. J. Balaban. An optimal algorithm for finding segments intersections. In *Proc. SoCG'95*, pages 211–219, 1995.

[2] P. Bose and P. Morin. Online routing in triangulations. *SIAM J. Comput.*, 33(4):937–951, 2004.

[3] E. Buyukkaya and M. Abdallah. Efficient triangulation for P2P networked virtual environments. In *Proc. NetGames'08*, pages 34–39, 2008.

[4] D. P. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete Comput. Geom.*, 5:399–407, 1990.

[5] J. Liebeherr, M. Nahas, and W. Si. Application-layer multicasting with Delaunay triangulation overlays. *IEEE J. Sel. Areas Comm.*, 20:1472 – 1488, 2002.

[6] C. L. Monma and S. Suri. Transitions in geometric minimum spanning trees. *Discrete Comput. Geom.*, 8:265–293, 1992.

Construction of Common Unfolding of a Regular Tetrahedron and a Cube

Toshihiro Shirakawa* Takashi Horiyama† Ryuhei Uehara‡

Abstract

A procedure that produces a common unfolding of a regular tetrahedron and a cube is given. It is adaptable to the length of an edge. If we allow a small error of the length of an edge of the tetrahedron, the procedure certainly halts and generates a common unfolding of a cube and an almost regular tetramonohedron. If we wish to generate the common unfolding of them with accuracy, we conjecture that the procedure does not halt and we obtain the common unfolding in the limit as a set of infinitely many points. The procedure has a potential to design a fractal structure given in a continued fraction form.

1 Introduction

Recently, several polygons that can fold to two different polyhedra have been developed (Table 1). Such a polygon is called a *common unfolding* of the polyhedra[1]. Observing these impressive unfoldings, it is natural to ask that whether there is a common unfolding of two (or more) different Platonic solids. This question has arisen several times independently, and it is still open (see [3, Section 25.8.3]).

In this paper, we give a procedure that generates a common unfolding of a regular tetrahedron and a cube[2]. The procedure is adaptable to the length of an edge of the tetrahedron. More precisely, the procedure generates the common unfolding of a cube and an almost regular tetramonohedron. If we wish to generate the common unfolding of a cube and a regular tetrahedron, the procedure produces a set of an infinite number of points, and we obtain the common unfolding in the limit. In a sense, this procedure gives an affirmative answer to the open problem; there exists a common unfolding of two Platonic solids (if our conjecture based on experiments is true). When we admit some error, say $\epsilon > 0$, then the procedure always halts and it certainly generates a common unfolding of a cube and an almost regular tetramonohedron whose edge lengths are within the interval $[\ell-\epsilon, \ell+\epsilon]$, where $\ell = \sqrt{2\sqrt{3}}$ is the length of an edge of the regular tetrahedron of surface area 6. Experimentally, we obtain a common unfolding of a cube and an almost regular tetramonohedron with $\epsilon < 2.89200 \times 10^{-1796}$.

Although we obtain an unfolding close to the ideal one, the connectivity of the unfolding generated by the procedure is not guaranteed in general. From the experimental results, we conjecture some useful properties of the unfolding. Based on it, we can take arbitrarily small $\epsilon > 0$ to obtain such an unfolding of a cube and an almost regular tetramonohedron.

The behavior of the procedure seems to rely on the continued fraction form of the length of an edge. On the other hand, it is well known that some real numbers have simple (but infinite) continued fraction forms. The conjecture would imply that our procedure is also useful to generate so called fractal curves based on these forms; see Figures 6 and 7 in Appendix.

2 Preliminaries

We first show some basic results about unfolding of a convex polyhedron.

Lemma 1 ([3, Sec. 22.1.3]) *All vertices of a polyhedron X are on the edges of any unfolding of X.*

Let P be a polygon on the plane, and R be a set of three points (called *rotation centers*) on the boundary of P. Then P has a *tiling* called symmetry group *p2* if P fills the plane by the repetition of 2-fold rotations around the points in R. The filling should contain no gaps nor overlaps. The rotation defines an equivalence relation on the points in the plane. Two points p_1 and p_2 are mutually equivalent if p_1 can be moved to p_2 by the 2-fold rotations. More on the properties of p2 tiling can be seen, e.g., in [5]. Based on the notion of p2 tiling, any unfolding of a tetramonohedron[3] can be characterized as follows:

Theorem 2 ([1, 2]) *P is an unfolding of a tetramonohedron if and only if (1) P has a p2 tiling, (2) four of the rotation centers consist in the triangular lattice formed by the triangle faces of the tetramonohedron, (3) the four rotation centers are the lattice points, and (4) no two of the four rotation centers belong to the same equivalent class on the tiling.*

*G-mode Co. Ltd., Japan, ZVB03747@nifty.ne.jp

†Graduate School of Science and Engineering, Saitama University, Japan, horiyama@al.ics.saitama-u.ac.jp

‡School of Information Science, JAIST, Japan, uehara@jaist.ac.jp

[1] Note that an edge of an unfolding can pass through a flat face of the polyhedra. See an unfolding of a cube in Figure 1.

[2] In this paper, a *cube* always means a unit cube that is a box of size $1 \times 1 \times 1$.

[3] A *tetramonohedron* is a tetrahedron that consists of four congruent triangles.

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

A regular octahedron and a tetramonohedron	[3, Figure 25.50]
A regular tetrahedron and a box of size $1 \times 1 \times \sqrt{3} - 1/2 = 1.232$	[3, Figure 25.51]
A cube and a tetramonohedron of size $1 : \sqrt{34}/6 : \sqrt{34}/6 = 0.9718$	[6]
A regular octahedron and a tetramonohedron of size $1.0072 : 0.9965 : 0.9965$	[6]
A regular icosahedron and a tetramonohedron of size $1 : 1.145 : 1.25$	[4]

Table 1: Known common unfoldings of a regular polyhedron and another (nonregular) polyhedron.

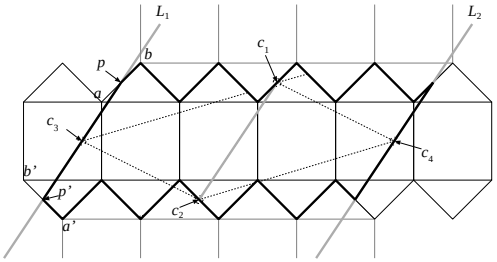


Figure 1: An initial unfolding P_1 of a cube.

We start from an unfolding P_1 of a cube in Figure 1. (The points c_1, c_2, p, p' are the centers of corresponding edges.) The thick lines in the figure gives an unfolding of a cube. Also, P_1 satisfies the conditions of Theorem 2 with four points c_1, c_2, c_3, c_4; that is, P_1 is an unfolding of a tetramonohedron. We here note that c_1c_2 is in parallel to the lines L_1 and L_2, and the tiling is split by lines L_1 and L_2 into "parallel ribbons." Thus we obtain infinitely many tilings by shifting these ribbons into any position along the lines. This fact implies that we can move the rotation centers c_3 and c_4 to *any* positions along L_1 and L_2, respectively, as long as $|c_1c_3| = |c_2c_4|$ (and hence $|c_1c_4| = |c_2c_3|$) without changing the surface area of the tetramonohedron. That is, P_1 is a common unfolding of a cube and an infinitely many tetramonohedra. (In the context of [3, Section 25], L_1 and L_2 make a *rolling belt* that is zipped to the edge c_3c_4.) Here we have $|c_1c_2| = \sqrt{13}/2 = 1.80278$, and each area of four congruent triangles is $3/2$. Thus taking $|c_3c_1| = |c_3c_2| = |c_4c_1| = |c_4c_2|$ in Figure 1, we have the following lemma:

Lemma 3 *There exists a common unfolding of a cube and a tetramonohedron with edge lengths* $\sqrt{13}/2 : \sqrt{745/208} : \sqrt{745/208} = 1.80278 : 1.89255 : 1.89255$.

The tetramonohedron in Lemma 3 is close to a regular tetrahedron. Our goal is to modify the edge lengths to the equal length $\sqrt{2\sqrt{3}} = 1.86121$.

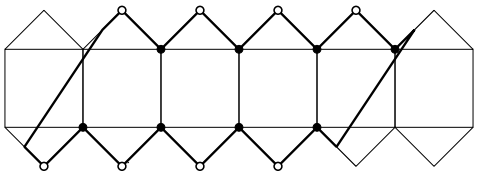


Figure 2: Fixed points on an unfolding.

3 Procedure for a common unfolding

Our procedure transforms P_1. More precisely, it moves c_1 to the right and c_2 to the left, respectively. Through these transformations, we keep two invariants for the polygon P_1 that P_1 is an unfolding of a cube, and P_1 is an unfolding of a tetramonohedron with $|c_1c_3| = |c_1c_4| = |c_2c_3| = |c_2c_4|$. Hence, when $|c_1c_2|$ becomes $\sqrt{2\sqrt{3}}$, we obtain a common unfolding of a cube and a regular tetrahedron. When we move c_1 and c_2, a series of discrete processes occurs. It will not terminate if we try to change from $|c_1c_2| = \sqrt{13}/2$ to $|c_1c_2| = \sqrt{2\sqrt{3}}$ as shown later. Thus we have two choices; the common unfolding of a cube and the regular tetrahedron in the limit as the set of infinitely many points, or a common unfolding of a cube and an *almost regular tetramonohedron* for any given error $\epsilon > 0$. Here we define the *almost regular tetramonohedron* with error $\epsilon > 0$ by a tetramonohedron with $|c_1c_2| \in [\sqrt{2\sqrt{3}} - \epsilon, \sqrt{2\sqrt{3}} + \epsilon]$.

Now we show how to stretch the distance between c_1 and c_2 and change $|c_1c_2|$ from $\sqrt{13}/2 = 1.80278$ to $\sqrt{2\sqrt{3}} = 1.86121$. Intuitively, we will slightly move the points c_1 and c_2 horizontally farther.

We here observe that, white circles in Figure 2 come to two "center points" in the top and bottom squares in the cube. If we remove these points, the squares have holes. On the other hand, if the unfolding contains these points inside, the squares have overlaps. Hence these points should be on the edge of P_1. Moreover, by Lemma 1, the vertices of the cube (black points in Figure 2) also should be on an edge of P_1. We call these immovable points *fixed points* of the unfolding.

Here we focus on the top edges of P_1. There are eight fixed points, and we have a rotation center c_1

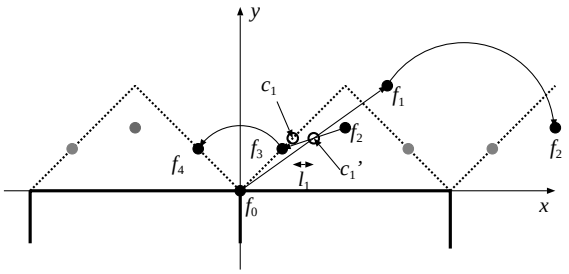


Figure 3: The construction of the points on edges of the unfolding of a cube and a tetramonohedron.

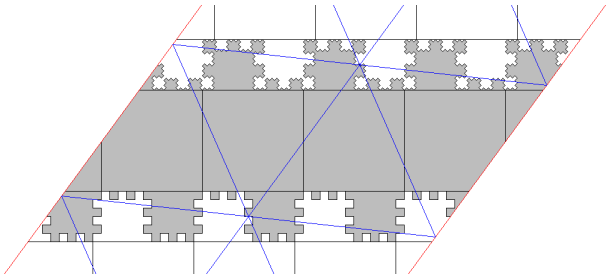


Figure 4: An example of an unfolding of a cube and a tetramonohedron.

on it. To describe exactly, we define xy-coordinate on the edges; let one of the fixed points, or the vertex of the (unit) cube closest to c_1 be $f_0 = (0, 0)$ (Figure 3). Then the initial position of c_1 is $(1/4, 1/4)$. Now, we assume that the rotation center $c_1 = (1/4, 1/4)$ is moved to $c_1' = (1/4+\ell_1, 1/4)$ for small ℓ_1. Since f_0 is a fixed point and c_1' should be the rotation center, $f_1 = (1/2 + 2\ell_1, 1/2)$ has to be a point on the edge of the unfolding. Moreover, when this unfolding folds up to a cube, f_1 will be put on the point $f_2 = (1+1/2, 1/2-2\ell_1)$. Hence f_2 is also a point on the edge of the unfolding[4]. Since they form a symmetrical unfolding, when the point $f_2 = (1+1/2, 1/2-2\ell_1)$ is placed, the equivalent point $(1/2, 1/2 - 2\ell_1)$ is also placed. That is, the equivalent relation $(x, y) \equiv (x - 1, y)$ is always applied. We can repeat this process and obtain a set of points which should be on edges of the unfolding with the new rotation center $c_1' = (1/4 + \ell_1, 1/4)$. In other words, we have a set of points that should be on edges of the unfolding with respect to the shift value ℓ_1. For this procedure, we have the following lemma:

Lemma 4 *The procedure of mapping the fixed points halts if and only if ℓ_1 is a rational number.*

Proof. If $\ell_1 = \frac{p}{q}$ with $0 < p < q$ for some positive integers p and q, all the mapped points from the fixed points are onto a lattice of size $O(pq)$. Hence the procedure can be terminated when the mapping visits some point again. On the other hand, if ℓ_1 is not a rational number, the same coordinate never appears and the procedure will not terminate. □

Using the procedure, we can slightly move c_1 and c_2 horizontally, and stretch $|c_1c_2|$ to the desired length. We now combine the construction in Figure 1; we also tilt the lines L_1 and L_2 in parallel with c_1c_2 consistent to the surface area, and obtain the desired unfolding. For example, Figure 4 is a common unfolding of a cube and a tetramonohedron such that $\ell_1 = 4/21$ and $\ell_2 = 5/24$, where ℓ_2 is the distance that c_2 is moved to the left.

In general, this procedure does not always generates a connected unfolding. More precisely, two lines L_1 and L_2 may cut the unfolding into disconnected pieces. To guarantee that these lines does not divide the unfolding, we have to investigate the generated edges by the procedure. We experimentally generated many unfoldings, and obtain the following observation, but we have no proof and no formal characterization of this sets of points up to now:

Observation 1 *We let $\ell_1 = (1 - \phi_1)/4$ and $\ell_2 = (1 - \phi_2)/4$, where ϕ_1 and ϕ_2 are given in continued fraction forms*[5] *$\phi_1 = \frac{1}{a_1\pm}\frac{1}{a_2\pm}\frac{1}{a_3\pm}\cdots\frac{1}{a_k}$, and $\phi_2 = \frac{1}{b_1\pm}\frac{1}{b_2\pm}\frac{1}{b_3\pm}\cdots\frac{1}{b_h}$. Then, each upper edge of the unfolding is given by the waves recursively defined by a_i as follows. We first replace a line segment of the original P_1 by a curve with a_1 "waves;" it is a triangular wave or a square wave depending on the parity of a_1. Then, each edge is again replaced by the waves decided by a_2, and so on. The signal determines the first direction of the wave.*

For example, in Figure 4, $\ell_2 = (1 - 1/6)/4 = 5/24$, or $\phi_2 = 1/6$. Hence the lower edges consist of a square wave of 6 peeks. On the other hand, since $\phi_1 = 5/21 = 1/(4 + 1/5)$, each upper edge consists of a square wave of four peeks of which each edge of the square consists of a triangular wave of five peeks.

We conjecture that the observation certainly holds, but no proof and no exact characterization of the "wave" are given. However, we can construct an accurate connected unfolding based on the observation. Precisely, we obtain several values by a brute force algorithm that alternately chooses $a_1, b_1, a_2, b_2, \ldots$ to

[4] Precisely, before computing this "rotation" of the point f_1 to f_2, we have to determine if f_1 is on the left side or on the right side of the unfolding. Depending on the side, we have to choose the direction of the rotation from clockwise and counterclockwise. When the point f_i is just on the boundary, or when $f_i = (0, j)$ or $f_i = (0.5, j)$ with $j > 0$, we need some "clue;" to decide that, we also maintain which side is "inside" of the unfolding for each point f_i. But the details are omitted in this draft.

[5] This "continued fraction form" is slightly different from the standard one. In the standard form, $a_i \geq 1$ and all signs are "+." In our continued fraction form, we use "−" and avoid the case $a_i = 1$ for $i > 1$. The details are omitted in this draft.

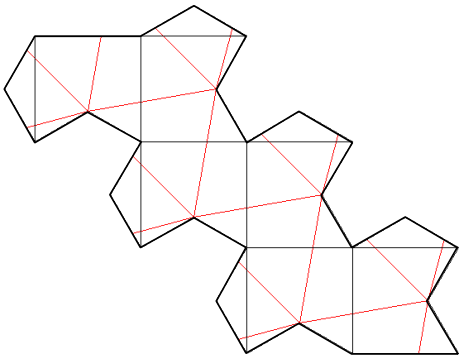


Figure 5: A common unfolding of a cube and a (nonregular) octahedron.

be closer $\ell_1 + \ell_2 = \sqrt{2\sqrt{3} - 9/4} - 1$: $a_1 = 4, b_1 = 6, a_2 = 6, b_2 = -34, a_3 = -42, b_3 = -14, a_4 = -116, b_4 = -2146, a_5 = 4010, b_5 = -3316, a_6 = -4958, b_6 = 8684, a_7 = -7820, b_7 = 7082, a_8 = 2668, b_8 = -3684, a_9 = 4564, b_9 = 1662, a_{10} = 560, b_{10} = -158, \ldots$. We have obtained a connected unfolding for each i. When we use up to a_{10} and b_{10}, we obtain $\epsilon < 4.63451 \times 10^{-56}$, and the values up to a_{50} and b_{50} give us $\epsilon < 2.89200 \times 10^{-1796}$:

Theorem 5 *There exists a common unfolding of a cube and an almost regular tetramonohedron with an error* $\epsilon < 2.89200 \times 10^{-1796}$.

We conjecture that this process can be repeated in arbitrary large, and we obtain a connected common unfolding of a cube and a regular tetrahedron in the limit:

Conjecture 1 *There exists a series of points that converges to a connected common unfolding of a cube and a regular tetrahedron in the limit.*

4 Concluding Remarks

We give a procedure that generates a common unfolding of a cube and a regular tetrahedron. But this is not completed: the procedure does not halt, and it might produce a disconnected unfolding. The proof of the observation, or the characterization of the curve generated by the procedure is a future work. We also conjecture that a similar construction of a common unfolding of a regular octahedron and a tetramonohedron works.

It is a challenging problem to try to the other pair of Platonic solids rather than a regular tetrahedron; in the case, we cannot use tiling as a tool any more. However, recently, the first author finds a common unfolding of a cube and a (nonregular) octahedron (Figure 5; note that this octahedron consists of two regular

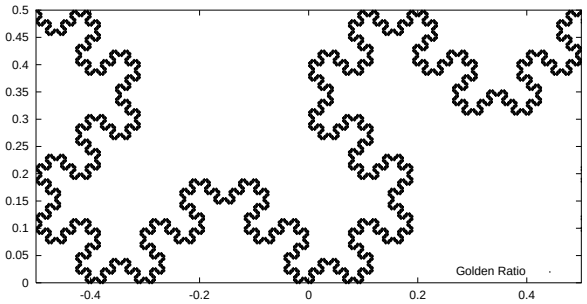


Figure 6: The set of the first 5000 points given by the golden ratio $\phi = \frac{1+\sqrt{5}}{2} = 1 + \frac{1}{1+}\frac{1}{1+}\frac{1}{1+}\cdots$.

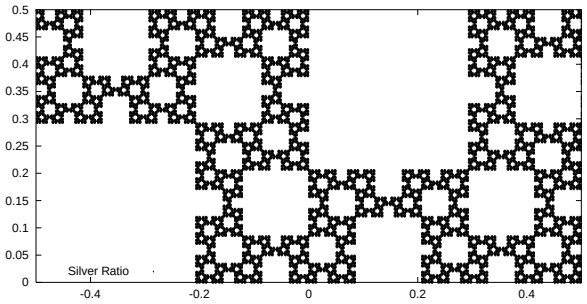


Figure 7: The set of the first 10000 points given by the silver ratio $\phi = \sqrt{2} - 1 = \frac{1}{2+}\frac{1}{2+}\frac{1}{2+}\cdots$.

triangles of edge length $\sqrt{2}$, and six triangles of edge lengths $\sqrt{2}, \sqrt{24 - 6\sqrt{3}}/3, \sqrt{24 - 6\sqrt{3}}/3)$. Hence it is not so easy to give some negative results, and we might have a common unfolding of the other solids.

Appendix

Based on the observation, we can design and generate some fractal pattern for a certain real number ϕ. Some experimental results are given in Figures 6 and 7.

References

[1] J. Akiyama. Tile-Makers and Semi-Tile-Makers. *American Mathematical Monthly*, 114:602–609, 2007.

[2] J. Akiyama and C. Nara. Developments of Polyhedra Using Oblique Coordinates. *J. Indonesia. Math. Soc.* , 13(1):99–114, 2007.

[3] E. D. Demaine and J. O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra.* Cambridge University Press, 2007.

[4] T. Horiyama and R. Uehara. Nonexistence of Common Edge Developments of Regular Tetrahedron and Other Platonic Solids. In *China-Japan Joint Conference on Computational Geometry, Graphs and Applications (CGGA 2010)*, 2010.

[5] D. Schattschneider. The plane symmetry groups: their recognition and notation. *American Mathematical Monthly*, 85:439–450, 1978.

[6] T. Shirakawa. Unpublished. 2010.

The Traveling Salesman Problem with Differential Neighborhoods

Lauri Häme* Esa Hyytiä* Harri Hakula*

Abstract

We introduce a novel differential approach to the Traveling Salesman Problem with Disk Neighborhoods (TSPDN), in which each node may be relocated within radius r from the original location in order to decrease the length of the shortest tour visiting all nodes. When r is small compared to the distance between the nodes, the optimal solution to the TSPDN is achieved by shortening the cycle corresponding to the optimal TSP tour without reordering the nodes. Looking at the shortening rate of a cycle, defined as the ratio of the decrease in tour length to r when r tends to zero, gives us an insight on how the movement of nodes can be converted into savings in tour length. We study the optimal direction for shortening and show how the shortening rate relates to the tightness of turns, number of U-turns and the distance from the origin in Euclidean, Manhattan and hyperbolic metrics, respectively.

1 Introduction

A natural generalization of the Traveling Salesman Problem (TSP) is the TSP with neighborhoods (TSPN): given a collection of n regions (disks, rectangles, lines, ...), the goal is to find a shortest tour that visits all of them [2, 5].

In this work, neighborhoods are defined as disks centered at the initial locations of nodes to be visited by the salesman and the radius r of the disks denotes the maximum distance each node can be shifted in order to decrease the total tour length. We may equivalently think that there are n customers and r denotes the maximum distance each customer can walk to meet the salesman. We are interested in studying the effect of relocating nodes on total tour length, by comparing the optimal TSP tour to the solution of the TSP with disk neighborhoods (TSPDN).

While the absolute difference between the solutions to TSP and TSPDN is clearly increased with r, the most efficient "steps" are taken when r is small. This is seen to be true also for arbitrary cycles, consisting of a finite number of nodes, that are shortened by shifting each node a distance r in the optimal direction. We examine the *shortening rate* of a cycle, defined as the ratio of the decrease in the length of the cycle to r when $r \to 0$. This approach is in fact similar to discrete curve shortening flow models presented in [3, 12, 16, 18]. Most of these works focus on the asymptotic behavior of polygons that evolve according to a specific shortening flow (see section 1.1). In contrast to these models, we assume that all nodes move at equal "speed", that is, r is equal for each node, and focus on the optimal direction for shortening and the corresponding maximum shortening rate. By shifting the nodes in an optimal TSP tour a differential distance dr in the optimal direction, we achieve an optimal solution to the TSP with differential neighborhoods.

The main contributions are Theorems 3, 4 and 5, which characterize the optimal direction for shortening and the shortening rate in arbitrary cycles in different metrics: The shortening rate is dependent on the sharpness of turns in the Euclidean metric, number of U-turns in the Manhattan metric and the distance from the origin in the hyperbolic metric. Furthermore, we derive global bounds for the shortening rate in arbitrary cycles and upper bounds for the shortening rate in optimal TSP tours.

This work is partially motivated by different types of vehicle routing problems, where the shortening rate translates to the available savings in mileage given the nodes (goods, passengers, customers of a mobile service provider) are shifted in optimal directions.

Proofs are omitted due to space limitations. However, we give a short description of each proof.

1.1 Related work

In the Group-TSP [7], also known as the One-of-a-Set-TSP [14] and the Errand Scheduling problem [17], a salesman has to meet n customers, to each of which is associated a set of at most k possible meeting places. If the weights are symmetric, the optimal solution can be polynomially approximated with ratio $3k/2$ [17].

The Group-TSP in which the neighborhoods are connected regions in the plane is referred to as TSP with neighborhoods (TSPN). In [11], a polynomial time algorithm with approximation ratio $O(n^2 \log n)$ for the general TSPN is provided, where n is the number of neighborhoods. In [2], constant factor approximation algorithms are presented for special cases in which the neighborhoods are defined as parallel unit segments, translations of convex polygonal neighborhoods and circles. These results are extended in [6] to neighborhoods with comparable diameter, unit disks, and infinite lines. In [11], a polynomial time approximation scheme (PTAS) is introduced for the special case where the tour is short compared to the size of the neighborhoods. In [15], a PTAS for the TSPN with disjoint fat regions in the plane is presented.

In addition to the TSPN, our approach is closely related to curve shortening flow [8, 9, 4], in the Euclidean version

*Aalto University School of Science, Finland. E-mail: lauri.hame@tkk.fi, esa.hyytia@tkk.fi, harri.hakula@tkk.fi

of which the length of a closed continuous curve is decreased by shifting each point on the curve along the normal vector at a rate proportional to curvature. This method shrinks the length of a curve as fast as possible using only local information [10].

The results obtained in the curve shortening literature have motivated research in creating discrete analogues of the flows [18]. Reference [3] introduces a linear polygon shortening flow where each vertex chases the centroid of its two neighboring vertices, which shrinks polygons to elliptical points. In addition, [3] proposes a Euclidean polygon shortening scheme based on the Menger-Melnikov curvature [13]. In [12] it is shown that in this case most quadrilaterals shrink to circular points. In [16], a discrete curve shortening equation is formulated such that the perimeter of the polygon is monotonically decreasing.

For the linear polygon shortening flow, proposed in [3], it is shown in [18] that 1) polygons shrink to elliptical points, 2) convex polygons remain convex, 3) if vertices are arranged in a star formation about their centroid, they will remain in a star formation for all time and 4) the perimeter of the polygon is a monotonically decreasing function of time. In addition, the authors derive the optimal direction for perimeter shortening.

2 Preliminaries

In the following, $S = (s_1, \ldots, s_n)$ denotes a cycle of n distinct nodes in a metric space (X, d) and $|S| = \sum_{i=1}^{n} d(s_{i-1}, s_i) \pmod n$ denotes the length of the cycle with respect to metric d. $S(P)$ denotes the cycle corresponding to the optimal solution to the classical traveling salesman problem with the node set $P = \{p_1, \ldots, p_n\}$. Moreover, $L_S(r)$ denotes the minimum length of all cycles $(q_1, \ldots, q_n)$ for which $d(q_i, s_i) \leq r$ for all i.

If the nodes of the problem are relatively far apart from each other compared to r, the TSPDN may be solved up to optimality by improving the optimal center tour (the solution to the TSP) without reordering nodes. In this case, we say that the set of nodes is r-stable.

It is easy to see that any fixed node set for which the minimum distance between two nodes is $\epsilon > 0$, will become r-stable when r is decreased (see Figure 1).

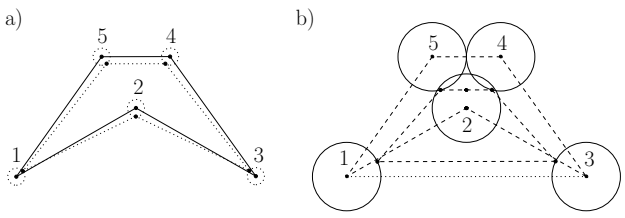


Figure 1: Stability of TSPDN. The solid lines represent the optimal solution to the classical TSP (center tour) and the dashed lines represent the optimal solution to the r-TSPDN. As r is decreased, the two solutions converge.

In this work, we focus on the difference between TSP and TSPDN when r tends to zero. Thus, the optimal ordering of the TSP is preserved.

3 Analysis

The shortening rate in a cycle S is defined by $-L'_S(0) = \lim_{r\to 0} \frac{L_S(0)-L_S(r)}{r}$, which is equal to the decrease in the length of the cycle, relative to the differential radius of the neighborhoods. $L'_S(0)$ corresponds to the change in the length of S when each customer takes a differential step dr in the optimal direction. Note that $-L'_S(0) \geq 0$, since $L_S(r) \leq L_S(0)$ due to the fact that the nodes of the center tour $L_S(0)$ are always included in the neighborhoods of radius $r \geq 0$. A large absolute value of $L'_S(0)$ means that the shortening rate is significant. In the following, we will study the shortening rate in arbitrary cycles with Euclidean, Manhattan and hyperbolic metrics.

3.1 Global bounds

Theorem 1 *Let $S = (s_1, \ldots, s_n)$ be a cycle in a metric space (X, d). Then, $L_S(0) - L_S(r) \leq 2nr$ for all $r \geq 0$.*

Theorem 1 states that no cycle can be shortened by more than than twice the total walking distance of customers. This is proved by adding a detour of at most $2r$ from each optimal meeting location to the original location. For the case studied in [6], in which the neighborhoods are defined as unit disks, the theorem can be applied as follows: For disjoint unit disks, Theorem 1 implies $L_S(0) - L_S(1) \leq 2n \leq L_S(0)$. By looking at the corresponding inequality derived from [6], namely $L_S(0) - L_S(1) \leq |L_S(0)|\frac{8+8\pi/L_S(0)}{\pi+8}$, it can be seen that our result improves this bound whenever the length $L_S(0)$ of the center tour satisfies $L_S(0) < 8$. For arbitrary unit disks, we derive from [6] the inequality $L_S(0) - L_S(1) \leq |L_S(0)|\frac{\pi+7+10\pi/L_S(0)}{\pi+8}$. Our upper bound $2n$ improves this result whenever $2n < L_S(0) < 10\pi$.

Consider next the corresponding lower bound for $L_S(0) - L_S(r)$. For all cycles visiting $n \geq 2$ disks centered on a straight line, of which at least two are disjoint, we have $L_S(0) - L_S(r) = 4r$. For the general case, we obtain the following result by considering a minimum bounding circle of radius R and scaling it down by $(1 - \frac{r}{R})$.

Theorem 2 *Let $S = (s_1, \ldots, s_n)$ be a cycle in $\mathbb{R}^2$ satisfying $\max_{i,j\in\{1,\ldots,n\}} \|s_i - s_j\| \geq 2r \geq 0$. The minimum length $L_S(r)$ satisfies $L_S(0) - L_S(r) \geq 4r$ in the Euclidean (L^2) and Manhattan (L^1) metrics.*

3.2 Euclidean metric

In [18], it has been shown that for a planar Euclidean cycle, the optimal *direction* for differential perimeter shortening is towards the bisector of the angle between two subsequent legs of the cycle. In the following, we examine the corresponding *shortening rate* assuming that all nodes move at equal speed in the optimal direction.

Theorem 3 *Let $S = (s_1, \ldots, s_n)$ be a cycle in $\mathbb{R}^2$ with angles α_i at corner s_i. In the Euclidean metric, the shortening rate of S is given by the formula $-L'_S(0) = 2\sum_{i=1}^{n} \cos\frac{\alpha_i}{2}$.*

This result, which is proved by considering a sequence of intermediate cycles, states that the shortening rate depends only on the angles between subsequent legs of a cycle. Assuming that the angles in a cycle S are uniformly distributed between 0 and π, the expected shortening rate in the cycle is given by $E(-L'_S(0)) = 2n\int_0^{\pi} \frac{1}{\pi}\cos\frac{\alpha}{2}d\alpha \approx 1.273n$. Furthermore, since $2\cos 60° = 1$, it can be stated that if the angle corresponding to a node i in a cycle is less than 120°, then the improvement in the length of the cycle achieved by relocating i is asymptotically greater than the walking distance r.

3.3 Manhattan metric (L^1-norm)

Definition 1 *A sequence of nodes $(s_h, s_{h+1}, \ldots, s_k)$ in a cycle $S = (s_1, \ldots, s_n) = ((x_1, y_1), \ldots, (x_n, y_n))$ (mod n) in $\mathbb{R}^2$ is a U-turn with respect to x if $x_h = x_{h+1} = \ldots = x_k$ and $\operatorname{sgn}(x_h - x_{h-1}) = \operatorname{sgn}(x_k - x_{k+1}) \neq 0$.*

A U-turn with respect to y is defined similarly. Two U-turns X and Y are called disjoint, if $X \cap Y = \emptyset$. The following theorem establishes a relation between the number of U-turns and the shortening rate.

Theorem 4 *Let $S = (s_1, \ldots, s_n)$ be a cycle in $\mathbb{R}^2$ and let M denote the minimum depth of U-turns in S. If $r < M/2$, the length of the cycle in the Manhattan metric satisfies $L_S(0) - L_S(r) = 2U(S)r$, where $U(S)$ is the maximum number of disjoint U-turns in S.*

This result is proved similarly as Theorem 3. For disjoint disk neighborhoods of radius r, the contribution of each U-turn on the decrease of the length of a cycle is $2r$. If all nodes are U-turns, a cycle can be shortened by $2nr$, which is an upper bound for all cycles. Differentiating the result yields the shortening rate $-L'_S(0) = 2U(S)$. For example, the expected shortening rate for a random cycle S in $\mathbb{R}^2$ equals $E(-L'_S(0)) = 2 \cdot \frac{3}{4} = \frac{3}{2}n$.

Clearly, the shortening rate is invariant to linear transformations since the number of U-turns depends only on the signs of the differences between subsequent nodes. More precisely, applying a linear function $f(x, y) = (a_x + b_x x, a_y + b_y y)$, where $b_x, b_y \neq 0$, on the points of any cycle will not affect the number of U-turns.

Example 1 *The expected number of U-turns in a random cycle S in $[0, a] \times [0, b] \in \mathbb{R}^2$, consisting of n nodes equals $E(U(S)) = \frac{n}{ab}\int_0^b \int_0^a P(U \mid x, y)\, dxdy = \frac{8}{9}n$. The expected shortening rate thus equals $E(-L'_S(0)) = \frac{16}{9}n \approx 1.78n$, independent of the ratio of a to b.*

For a random cycle S in an ellipse with half axes a and b, we get $E(-L'_S(0)) = \frac{n}{6}\left(11 - \frac{2}{\pi^2}\right) \approx 1.80n$. Again, $E(-L'_S(0))$ is independent of the eccentricity of the ellipse.

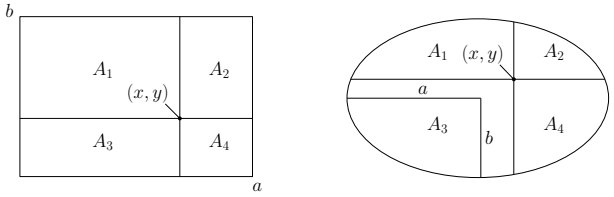


Figure 2: Illustration of Example 1. The probability that a node located in (x, y) is a U-turn in a random cycle is given by $P(U \mid x, y) = 1 - \frac{2}{(A_1+A_2+A_3+A_4)^2}(A_1A_4 + A_2A_3)$.

3.4 Hyperbolic metric

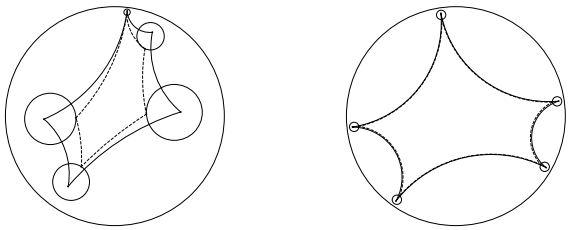


Figure 3: The TSPDN in the Poincaré disk model. The solid and dashed lines represent the optimal center tour and a solution to the TSPDN, respectively.

In hyperbolic geometry, all postulates of Euclidean geometry are satisfied except the parallel postulate. In the Poincaré hyperbolic disk [1], the hyperbolic distance between two points u, v in the unit disk is defined by the formula $d(u, v) = \operatorname{arccosh}\left(1 + \frac{2\|u-v\|^2}{(1-\|u\|^2)(1-\|v\|^2)}\right)$, where $\|\cdot\|$ denotes the Euclidean norm. If $\|u\|$ and $\|v\|$ are small, $d(u, v)$ can be approximated by $2\|u - v\|$. The hyperbolic disk neighborhood with hyperbolic radius $\rho \geq 0$ of a node u consists of points v for which $d(u, v) \leq \rho$. Note that the hyperbolic disk corresponds to a disk in the Euclidean plane, but the Euclidean center of the disk is different from the hyperbolic center u unless $u = (0, 0)$ (see reference [1]). Let us consider a solution to the hyperbolic TSPDN obtained by shifting each node in the optimal center tour an equal hyperbolic distance ρ towards the origin (see Figure 3), which gives us a lower bound for the shortening rate.

Theorem 5 *Let $S = (s_1, \ldots, s_n)$ be a cycle in the unit disk and $r_i = \|s_i\|$ for $i \in \{1, \ldots, n\}$, where $\|\cdot\|$ denotes the Euclidean metric. The shortening rate of S in the hyperbolic metric satisfies $-L'_S(0) \geq \sum_{i=1}^{n}(r_i + r_{i+1})\left(1 - \frac{|r_i - r_{i+1}|}{\|s_i - s_{i+1}\|}\right)$, where $s_{n+1} = s_1$ and $r_{n+1} = r_1$.*

This result is proved by differentiating the hyperbolic length of leg (s_i, s_{i+1}) with respect to r. If all nodes are located at an equal Euclidean distance x from the origin, we have $-L'_S(0) \geq \sum_{i=1}^{n} 2x = 2xn$. Since by Theorem 1, the shortening rate always satisfies $-L'_S(0) \leq 2n$, our approximate solution is asymptotically optimal when the nodes approach the border of the unit disk.

Corollary 6 *Let $S_2, S_3, \ldots$ be a sequence of cycles of n nodes, where $S_j = ((1-1/j, \theta_1), \ldots, (1-1/j, \theta_n))$ in polar coordinates for all $j \geq 2$. Then, $\lim_{j\to\infty} -L'_{S_j}(0) = 2n$.*

3.5 Bounds for TSP tours

The shortening rate is generally governed by the tightness and number of turns in a cycle. In some cases, even the optimal TSP tour includes relatively tight turns. We state the following for Euclidean TSP tours in the plane.

Conjecture 1 *Let P be a node set in $\mathbb{R}^2$ involving $n > 2$ customers. The length of the shortest tour $S(P)$ satisfies $L_{S(P)}(0) - L_{S(P)}(r) \leq \sqrt{3}nr$ in the Euclidean metric.*

While the inequality in Conjecture 1 has not been proven to be valid, there exist no known problems P for which $-L'_{S(P)}(0) > \sqrt{3}n$. The equality is achieved for $n = 3$ with an equilateral triangle. In addition, it is possible to construct a sequence of problems that asymptotically satisfies the equality. For Manhattan TSP tours in the plane, the upper bound is equal to that of arbitrary cycles.

Theorem 7 *There exists (i) a sequence of problems $P_4, P_6, P_8, \ldots$ such that $\lim_{k\to\infty} \frac{1}{2k}(-L'_{S(P_{2k})}(0)) = \sqrt{3}$ in the Euclidean metric and (ii) for any $r \geq 0$ a sequence of problems $P_4, P_{12}, P_{20}, \ldots$ such that $L_{S(P_n)}(0) - L_{S(P_n)}(r) = 2nr$ for all $n = 4(2k-1), k \in \mathbb{N}$ in the Manhattan metric, where $|P_j| = j$ for all $j \in \mathbb{N}$.*

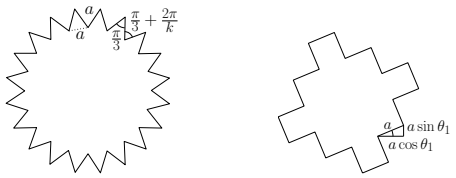


Figure 4: Theorem 7. The left figure shows a structure in which the shortening rate per node in the optimal Euclidean solution approaches $\sqrt{3}$ as $n \to \infty$. The right figure shows an optimal Manhattan TSP tour where each node is a U-turn and the cycle can be shortened by $2nr$.

Theorem 7 suggests that optimal Manhattan TSP tours may be shortened more efficiently than Euclidean TSP tours. In the hyperbolic metric, we know by Corollary 6 that for any sequence of cycles where the nodes approach the border of the unit disk, the shortening rate converges to $2n$. In this case, our solution approach to the hyperbolic TSPDN converges to the optimal solution.

4 Conclusions

In this work we study the difference between the length of an optimal TSP tour and the length of the optimal solution to the TSP with disk neighborhoods, in which each node can be redirected to a new location within a certain radius r from the original location. We show that the shortening rate is characterized by the tightness of turns in the Euclidean metric and it is equal to two times the number of U-turns with respect to the coordinate axes in the Manhattan metric. In the hyperbolic metric, the shortening rate increases with the distance of nodes from the origin.

References

[1] J. W. Anderson. *Hyperbolic Geometry*. Springer, 2005.

[2] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55:197–218, 1994.

[3] A. M. Bruckstein, G. Sapiro, and D. Shaked. Evolution of planar polygons. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(6):991 – 1014, 1995.

[4] K.-S. Chou and X.-P. Zhu. *The Curve Shortening Problem*. Chapman and Hall/CRC, 2001.

[5] M. de Berg, J. Gudmundsson, M. J. Katz, C. Levcopoulos, M. H. Overmars, and A. F. van der Stappen. TSP with neighborhoods of varying size. *Journal of Algorithms*, 57(1):22–36, 2005.

[6] A. Dumitrescu and J. S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *In Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 38–46, 2001.

[7] K. Elbassioni, A. V. Fishkin, N. H. Mustafa, and R. Sitters. Approximation algorithms for euclidean group TSP. In *Lecture Notes in Comp. Sci.* Springer Berlin, 2005.

[8] M. E. Gage. Curve shortening makes convex curves circular. *Inventiones Mathematicae*, 76:357 – 364, 1984.

[9] M. A. Grayson. The heat equation shrinks embedded plane curves to round points. *J. Diff. Geom.*, 26:285 – 314, 1987.

[10] M. A. Grayson. Shortening embedded curves. *Annals of Mathematics*, 129:71 – 111, 1989.

[11] J. Gudmundsson and C. Levcopoulos. A fast approximation algorithm for TSP with neighborhoods. *Nordic Journal of Computing*, 6(4):469–488, 1999.

[12] T. Jecko and J.-C. Leger. Polygon shortening makes (most) quadrilaterals circular. *Bull. Korean Math. Society*, 39(1):97 – 111, 2002.

[13] M. S. Melnikov. Analytic capacity: discrete approach and curvature of measure. *SB MATH*, 186(6):827 – 846, 1995.

[14] J. Mitchell. Geometric shortest paths and network optimization. In *J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry*. Elsevier, Amsterdam, 2000.

[15] J. S. B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. In *SODA '07*, pages 11–18. Society for Industrial and Applied Mathematics, 2007.

[16] K. Nakayama, H. Segur, and M. Wadati. A discrete curve-shortening equation. *Methods and Applications of Analysis*, 4(2):162 – 172, 1997.

[17] P. Slavik. The errand scheduling problem. *Tech. report 97-02, Dept. of Comp. Sci. and Engin., SUNY Buffalo*, 1997.

[18] S. L. Smith, M. E. Broucke, and B. A. Francis. Curve shortening and its application to multi-agent systems. In *CDC-ECC '05, Seville, Spain*, pages 2817 – 2822, 2005.

Design of Antigravity Slopes for Visual Illusion

Kokichi Sugihara*†

Abstract

This paper presents a method for designing solid shapes containing slopes where the orientation appears opposite to the actual orientation when observed from a unique vantage point. The resulting solids generate a new type of visual illusion, which we call "impossible motion", in which balls placed on the slopes appear to roll uphill, thereby defying the law of gravity. This is possible, because a single retinal image lacks depth information and human visual perception tries to interpret images as the most familiar shape, even though there are infinitely many possible interpretations. We specify the set of all possible solids represented by a single picture as the set of a system of equations and inequalities, and then relax the constraints in such a way that the antigravity slopes can be reconstructed. We present this design procedure with examples.

1 Introduction

This paper presents a computational approach to the design of a new visual illusion. Visual illusion is a perceptual behavior where what we "see" differs from the physical reality. This phenomenon is important in vision science because it helps us to understand the basis of human perception [5, 6]. Numerous traditional visual illusions are known, most of which are generated by two-dimensional pictures and their associated spatial manipulations [4, 10].

However, very few visual illusions are known that make use of three-dimensional solid shapes. An early example was the Ames room, where a person appears to become taller when moving from one corner of the room to another [3]. Other examples include impossible solids produced using a hidden-gap trick [2], and others without hidden gaps [8]. This latter classification was extended to include a new type of illusion called "impossible motion" [9].

Design of illusions using solids requires the application of mathematics, because this process can be counterintuitive.

This paper concentrates on one class of such solids called "antigravity slopes", in which balls appear to roll uphill against the law of gravity and produce the appearance of "impossible motion".

In Section 2, we specify the set of all possible solids represented by a picture, and in Section 3 we show that antigravity slopes cannot be constructed using that formulation. In Section 4, we remove some of the constraints, so design of antigravity slopes becomes possible. We show some examples in Section 5, and provide concluding remarks in Section 6.

2 Reconstruction of a Solid from a Picture

In this paper we consider polyhedrons, which are solids bounded by planar faces. We do not consider solids with curved surfaces. As shown in Fig. 1, we assume that a viewpoint is fixed at the origin of an xyz coordinate system, and that a picture of a solid is fixed at the plane $z = 1$.

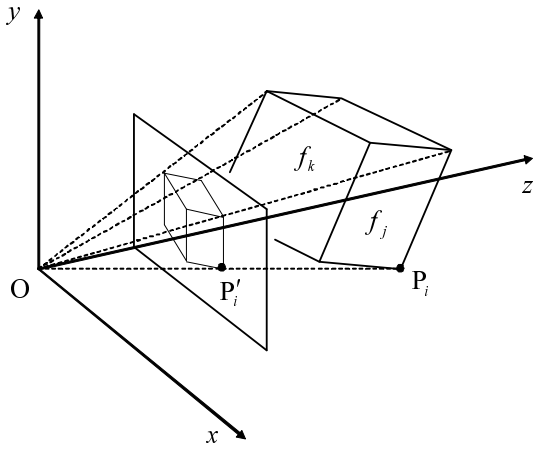


Figure 1: Central projection of a solid.

Suppose that we are provided with the relative relations among the vertices and the faces of the solid. Let $V = \{v_1, v_2, \ldots, v_n\}$ and $F = \{f_1, f_2, \ldots, f_m\}$ be the set of vertices and that of faces of the solid, respectively. Let $\mathrm{ON}(v_i, f_j)$ represent a predicate stating that the vertex v_i is on the face f_j. Similarly, let $\mathrm{NEARER}(v_i, f_j)$ indicate that "v_i is nearer to the viewpoint than the plane containing f_j", while $\mathrm{FARTHER}(v_i, f_j)$ indicates that "v_i is farther away than the plane containing f_j".

We assume that, in addition to the picture, we are given all predicates satisfied by the solid shape and

*Meiji Institute for Advanced Study of Mathematical Sciences, Meiji University. kokichis@isc.meiji.ac.jp
†JST, CREST

that we are interested in determining the reconstructability of a solid from the picture.

Let $(x_i, y_i, 1)$ be the coordinates of the i-th vertex on the picture plane. The original vertex on the solid should be on the half-line emanating from the origin that passes through the associated vertex on the picture plane, so we can express the coordinates of the original vertex in the space as $(x_i/t_i, y_i/t_i, 1/t_i)$, where t_i is an unknown parameter representing the inverse of the depth of the vertex from the viewpoint measured along the z axis.

Let

$$a_j x + b_j y + c_j z + 1 = 0 \tag{1}$$

be the plane containing the j-th face, where a_j, b_j and c_j are unknown.

Suppose that $\mathrm{ON}(v_i, f_j)$ is true. We can substitute the coordinates of v_i into the equation of f_j, and we find

$$a_j x_i + b_j y_i + c_j + t_i = 0, \tag{2}$$

which is linear for the unknowns. Similarly, if $\mathrm{NEARER}(v_i, f_j)$ is true, we find

$$a_j x_i + b_j y_i + c_j + t_i < 0, \tag{3}$$

and if $\mathrm{FARTHER}(v_i, f_j)$ is true, we find

$$a_j x_i + b_j y_i + c_j + t_i > 0. \tag{4}$$

We combine all equations of the form (2) for each ON predicate, and denote the resulting system of equations as

$$Aw = 0, \tag{5}$$

where $w = (z_1, \ldots, z_n, a_1, b_1, c_1, \ldots, a_m, b_m, c_m)^{\mathrm{t}}$ is the vector of unknown variables (t represents the transpose) and A is a constant matrix. Similarly, we collect all inequalities of the forms (3) and (4), and denote the resulting system of inequalities as

$$Bw > 0, \tag{6}$$

where B is a constant matrix, and the inequality symbol "$>$" represents componentwise inequalities, each of which is either "$>$" or "$<$".

We can prove that the picture represents a three-dimensional solid if and only if the system consisting of (5) and (6) has solutions [7]. Furthermore, the system of (5) and (6) is sensitive to errors of vertex positions in the picture, but a robust method for reconstructing the solid from the picture was also established [7].

3 Impossibility of Antigravity Slopes

We concentrate on antigravity slopes as a typical class of impossible motions. Let us consider the picture of a simple solid shown in Fig. 2, in which a slope is supported by two columns standing on a base plate.

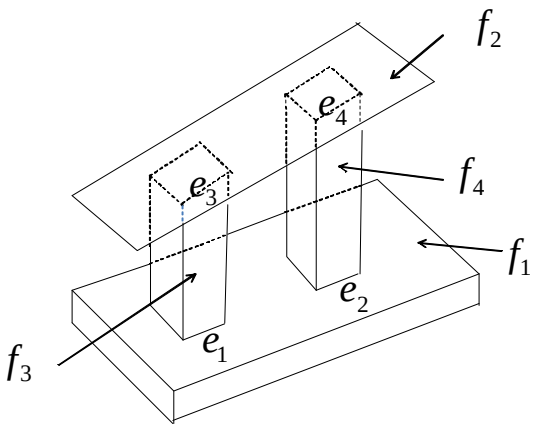


Figure 2: Slope supported by two columns.

The broken lines represent the hidden parts. However, to avoid unnecessary complexity, some hidden parts are not shown.

Let f_1 denote the top face of the base plate, and f_2 denote the slope plane. For each of the two columns, we assume that all four lower vertices are on f_1 and are farther away than f_2, while all four upper vertices are on f_2 and are nearer than f_1. We also assume that f_1 is horizontal. Thus, we expect that the slope f_2 tilts to the left, that is, the right end of f_2 is higher than the left end. Indeed, the system of equations (5) and inequalities (6) accepts such a slope as its solution.

Now we ask whether the set of the solutions contains a slope that tilts to the right. The answer is "no". In every solid whose projection matches that of the picture shown in Fig. 2, the slope f_2 tilts to the left. This can be understood in the following way. As shown in Fig. 2, let f_3 and f_4 be the right front faces of the left and right columns, e_1 and e_2 the lower edges of f_3 and f_4, and e_3 and e_4 the upper edges of f_3 and f_4, respectively. Because the edges e_1 and e_2 are collinear in the picture plane, and they are on f_1, then they must also be collinear in three-dimensional space. Let l_1 be a line in the space containing e_1 and e_2. e_3 and e_4 are collinear in the picture plane and they are on f_2, so they are collinear in three-dimensional space. Let l_2 be the line containing e_3 and e_4. Note that e_1 and e_3 are coplanar because they are on f_3; thus, l_1 and l_2 are coplanar, which implies that f_3 and f_4 are coplanar. l_1 and l_2 meet to the left of the solid, so the slope f_1 tilts to the left.

This property holds for any solution of the system of (5) and (6) associated with the picture shown in Fig. 2. Therefore, it is impossible to construct a slope that tilts to the right from the picture shown in Fig. 2.

4 Construction of an Antigravity Slope

Our goal is to construct a slope that tilts to the right, even though such a slope is not contained in the solutions of (5) and (6). Thus, we must modify the picture in such a way that the visible part of the solid does not change in the picture plane and the solutions of (5) and (6) contain a slope tilting to the right. To achieve this, we can modify the picture around the upper parts of the two columns because they are hidden by the slope. The vertices at the top of the columns can be moved slightly along the associated vertical edges of the columns. Here "slightly" means that the movements of the vertices are restricted to the area covered by f_2 in the picture plane.

Let v_i be one of the four top vertices of the left or right column, and let e_j be the vertical edge of the column incident to v_i. Let $(\alpha_j, \beta_j, 0)$ be the unit vector parallel to e_j in the picture plane. We replace the coordinates $(x_i, y_i, 1)$ of the vertex v_i with

$$(x_i + s_i\alpha_j, y_i + s_i\beta_j, 1), \tag{7}$$

where s_i is a new unknown parameter. Then, instead of the equation (2), the predicate $\mathrm{ON}(v_i, f_j)$ is represented by

$$a_j(x_i + s_i\alpha_j) + b_j(y_i + s_i\beta_j) + c_j + t_i = 0. \tag{8}$$

The inequalities of the forms (3) and (4) are also modified by replacing x_i and y_i with $x_i + s_i\alpha_j$ and $y_i + s_i\beta_j$, respectively. We change the equations and inequalities associated with all the upper vertices of the columns, and denote the resulting equations and inequalities as

$$\bar{A}(s)w = 0, \tag{9}$$

$$\bar{B}(s)w > 0, \tag{10}$$

where $s = (s_1, s_2, \ldots, s_k)$ is the vector of unknown parameters introduced by the movement of the hidden vertices (k denotes the number of hidden vertices to be moved), and $\bar{A}(s)$ and $\bar{B}(s)$ are the resulting coefficient matrices corresponding to the equations (5) and the inequalities (6).

The new system of equations (9) and (10) allows a solution corresponding to a slope tilting to the right, that is, an antigravity slope.

However, (9) and (10) are nonlinear because the matrices $\bar{A}(s)$ and $\bar{B}(s)$ contain unknown variables. Thus, unlike the system of (5) and (6), it is not straightforward to specify the set of all solutions. To overcome this difficulty, we employ the following convention instead of solving (9) and (10) directly.

Our solution is explained using the example shown in Fig. 2. Let v_1, v_2, v_3, v_4 be the top vertices of the left column, and v_5, v_6, v_7, v_8 be the top vertices of the right column. Thus, we have

$$\mathrm{ON}(v_i, f_2), \quad i = 1, 2, \ldots, 8 \tag{11}$$

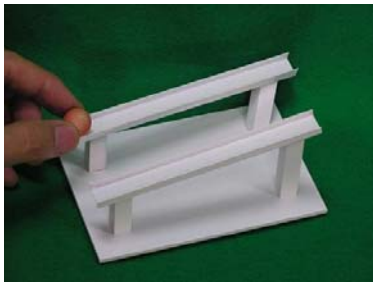
from the original solid structure. From these we accept two predicates,

$$\mathrm{ON}(v_1, f_2) \quad \text{and} \quad \mathrm{ON}(v_5, f_2), \tag{12}$$

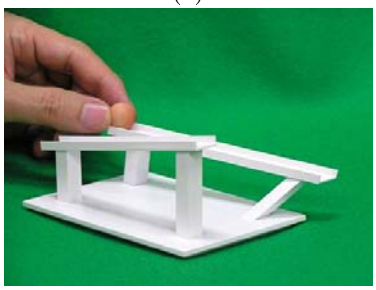
but ignore the other six, and reconstruct the equations (5) and the inequalities (6). As this system produces solutions in which slopes tilt to the right, we choose one of these slopes. In this solid, the six vertices $v_2, v_3, v_4, v_6, v_7, v_8$ are not necessarily on f_2, because the associated equations were ignored. We move these vertices in the three-dimensional space along their associated edges such that they are on f_2, until we obtain a solid in which all the original incidence relations are satisfied. In this solid, some of the vertices move from their original positions, but they are hidden in the slope and movements are restricted on the extended part of the visible edges. Thus, the visible part of the solid remains the same as that shown in the original picture. This describes our method for constructing antigravity slopes.

5 Examples

Figs 3, 4 and 5 show examples of antigravity slopes constructed by this method. In each figure, (a) shows a solid that looks the same as that represented by the original picture, but the orientations of the slopes are perceived as opposite to the actual orientation, and (b) shows the same solid seen from another angle.



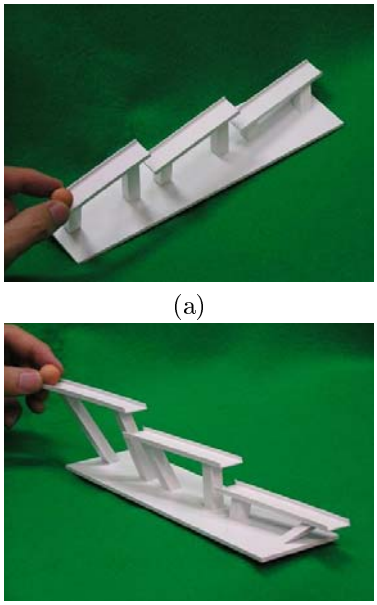
(a)



(b)

Figure 3: "Antigravity parallel slopes".

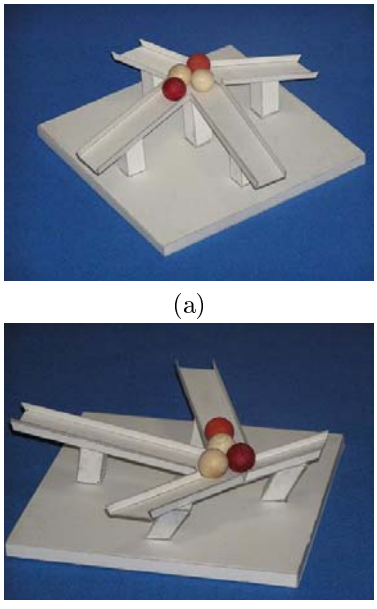
These examples generate impossible motions in that when we place balls on the slopes they appear to roll uphill against the slope, thereby defying the law of gravity. The impossible motion generated by



(a)

(b)

Figure 4: "Antigravity cascade of three slopes".



(a)

(b)

Figure 5: "Magnet-like slopes."

the solid shown in Fig. 5 was awarded first prize in the 2010 Best Illusion of the Year Contest held in Florida in May 2010 [1].

6 Concluding Remarks

We described our basic method for constructing antigravity slopes. When we observe antigravity slopes from a specific viewpoint, the orientations of the slopes appear opposite to the actual orientations, which generates a visual illusion of the impossible motion of rolling balls. This is a new computational approach to producing visual illusions. Future tasks include increasing the number of antigravity slope variants, extension to other types of impossible motion and the extension of the method to solids with curved faces. We intend to study human visual perception through visual illusions of impossible motion. We will address these basic research questions in future studies.

We also aim to increase possible applications for antigravity slopes, by developing methods for decreasing the strength of the illusion. It is known that one of the reasons for natural congestion of traffic flows on highways is driver misperception of slope orientation. If we can understand the mechanisms of human illusion in slope perception, we might inform the shape of new highways in which the true orientation of slopes can be easily perceived. We might also suggest possible methods for manipulating the environment around existing highways such that the illusion of slope is prevented.

Acknowledgments

This work is partially supported by the Grant-in-Aid for Scientific Research (B) No. 20360044 of MEXT.

References

[1] http://illusioncontest.newralcorrelate.com/ Web page of the Best Illusion of the Year Contest 2010.

[2] B. Ernst. The Eye Beguiled: Optical Illusion. Benedikt Taschen, 1986.

[3] R. L. Gregory. *The Intelligent Eye.* Weidenfeld & Nicolson, London, 1970.

[4] J. Ninio. *The Science of Illusions.* Comstock Pub. Assoc., New York, 2001.

[5] J. O. Robinson. *The Psychology of Visual Illusion.* Dover Publications, New York, 1998.

[6] A. Seckel. *Optical Illusions: The Science of Visual Perception.* Firefly Books Ltd., New York, 2009.

[7] K. Sugihara. *Machine Interpretation of Line Drawings.* MIT Press, Cambridge,1986.

[8] K. Sugihara. *Three-dimensional realization of anomalous pictures— An application of picture interpretation theory to toy design.* Pattern Recognition, vol. 30 (1997), pp. 1061–1067.

[9] K. Sugihara. *A characterization of a class of anomalous solids.* Interdisciplinary Information Sciences, vol. 11 (2005), pp. 149–156.

[10] J. Timothy Unruh. *Impossible Objects: Amazing Optical Illusions to Confound and Astound.* Sterling Publishing Co., Inc., New York, 2001.

Two-Site Voronoi Diagrams under Geometric Distance Functions

Gill Barequet* Matthew T. Dickerson† David Eppstein‡ David Hodorkovsky§ Kira Vyatkina¶

Abstract

We revisit a new type of a Voronoi diagram, in which distance is measured from a point to a *pair* of points. We consider a few more such distance functions, and analyze the structure and complexity of the nearest- and furthest-neighbor Voronoi diagrams of a point set with respect to these distance functions.

1 Introduction

The Voronoi diagram is a beautiful geometric structure, which has a wide variety of applications in the most diverse areas. Detailed surveys of its history, applications, and variants are given by Aurenhammer [4] and by Okabe, Boots, and Sugihara [11]. One of the recent generalizations of this concept is a family of so-called *2-site Voronoi diagrams* [5], which are based on distance functions that define a distance from a point in the plane to a *pair* of sites from a given set S. Consequently, each Voronoi region corresponds to an (unordered) pair of sites from S.

For S being a set of points, Voronoi diagrams under a number of 2-site distance functions have been investigated, which include arithmetic combinations of point-to-point distances [5, 13] and certain geometric distance functions [5, 7, 8]. In this work, we develop further the latter direction.

Let $S \subset \mathbb{R}^2$, and consider $p, q \in S$ and a point v in the plane. We shall focus our attention on a few circle-based distance functions:

- *radius of circumscribing circle*:
  $\mathcal{C}(v,(p,q)) = \mathrm{Rad}(\circ(v,p,q))$, where $\circ(v,p,q)$ is the circle defined by v, p, q,
- *radius of containing circle*:
  $\mathcal{K}(v,(p,q)) = \mathrm{Rad}(C(v,p,q))$, where $C(v,p,q)$ is the minimum circle containing v, p, q,
- *view angle*:
  $\mathcal{V}(v,(p,q)) = \measuredangle pvq$, or, equivalently, half of the angular measure of the arc of $\circ(v,p,q)$ that the angle $\measuredangle pvq$ subtends,

and on a parameterized perimeter distance function:

- *parameterized perimeter*:
  $\mathcal{P}_c(v,(p,q)) = |vp| + |vq| + c \cdot |pq|$, where $c \geq -1$.

The first and the third circle-based distance functions were first mentioned in [9]. The last function generalizes the perimeter distance function $\mathcal{P}(v,(p,q)) = \mathrm{Per}(\triangle(v,p,q)$ introduced in [5], and later addressed in [7, 8].

Since two points define a segment, any 2-point site distance function $d(v,(p,q))$ provides a distance between the point v and the segment pq, and vice versa. Consequently, geometric structures akin to 2-site Voronoi diagrams can arise as Voronoi diagrams of segments. This alternative approach was independently undertaken by Asano et al., and the "view angle" and the "radius of circumscribing circle" distance functions reappeared in their works [2, 3] on Voronoi diagrams for segments soon after they had been proposed by Hodorkovsky [9] in the context of 2-site Voronoi diagrams. However, as Asano's et al. research was originally motivated by mesh generation and improvement tasks, they were mostly interested in sets of segments representing edges of a simple polygon, and thus, non-intersecting (except, possibly, at the endpoints), what significantly alters the essence of the problem.

In this paper, we analyze the structure and complexity of 2-site Voronoi diagrams under the distance functions listed above. Our obtained results are mostly of theoretical interest. The method used to derive an upper bound on the complexity of the 2-site Voronoi diagram under the "parameterized perimeter" distance function is first developed for the case of $c = 1$, yielding a much simpler proof for the "perimeter" function than the one developed in [8], and then generalized to any $c \geq 0$.

Throughout the paper we use the notation $V_{\mathcal{F}}^{(n)}(S)$ (resp., $V_{\mathcal{F}}^{(f)}(S)$) for denoting the nearest- (resp., furthest-) 2-site Voronoi diagram, under the distance function $\mathcal{F}$, of a point set S. The set S is always assumed to contain n points.

2 Circumscribing Circle

Let $\circ(p,q,r)$ denote the unique circle defined by three distinct points p, q, and r in the plane.

*Dept. of Computer Science, The Technion—Israel Institute of Technology, Haifa 32000, Israel. E-mail: barequet@cs.technion.ac.il

†Dept. of Mathematics and Computer Science, Middlebury College, Middlebury, VT 05753. E-mail: dickerso@middlebury.edu

‡Dept. of Information and Computer Science, University of California, Irvine, CA 92717. E-mail: eppstein@ics.uci.edu

§Dept. of Applied Mathematics, The Technion—Israel Institute of Technology, Haifa 32000, Israel. Currently affiliated with Imagine Communications, Ltd. E-mail: david@imagine-com.com

¶Dept. of Mathematics and Mechanics, Saint Petersburg State University, 28 Universitetsky pr., Stary Peterhof, St. Petersburg 198504, Russia, E-mail: kira@math.spbu.ru, and Dept. of Natural Sciences, Saint Petersburg State University of Information Technologies, Mechanics and Optics, 49 Kronverkskiy pr., St. Petersburg 197101, Russia.

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

Definition 1 *Given two points p, q, the "circumcircle distance" $\mathcal{C}$ from a point v to the unordered pair (p, q) is defined as $\mathcal{C}(v, (p, q)) = \mathrm{Rad}(\circ(v, p, q))$.*

For a fixed pair of points p, q, the curve $\mathcal{C}(v, (p, q)) = \infty$ is the line $\overline{pq}$. This implies that all the points on $\overline{pq}$ belong to the region of (p, q) in $V_{\mathcal{C}}^{(f)}(S)$. In this section we assume that the points in S are in general position, i.e., there are no three collinear points, and no three pairs of points define three distinct lines that intersect at one point. The given sites are singular points, i.e., for any two sites p, q, the function $\mathcal{C}(v, (p, q))$ is not defined at $v = p$ or $v = q$.

Theorem 1 *Let S be a set of n points. The complexity of $V_{\mathcal{C}}^{(f)}(S)$ is $\Omega(n^4)$.*

Proof. The n points define $\Theta(n^2)$ lines with $\Theta(n^4)$ intersection points. All these intersection points are features of $V_{\mathcal{C}}^{(f)}(S)$, and hence the lower bound. □

Theorem 2 *Let S be a set of n points. The complexity of both $V_{\mathcal{C}}^{(n)}(S)$ and $V_{\mathcal{C}}^{(f)}(S)$ is $O(n^{4+\varepsilon})$ (for any $\varepsilon > 0$).*

Proof. The complexity of $V_{\mathcal{C}}^{(n|f)}(S)$ is identical to that of the respective diagram of $\mathcal{C}^2(v, (p, q)) = \mathrm{Rad}^2(\circ(v, p, q))$. It is known that $\mathrm{Rad}^2(\circ(v, p, q)) = ((|vp||vq||pq|)/(4|\triangle vpq|))^2 = (((v_x - p_x)^2 + (v_y - p_y)^2)((v_x - q_x)^2 + (v_y - q_y)^2)((p_x - q_x)^2 + (p_y - q_y)^2))/(4(v_x(p_y - q_y) - p_x(v_y - q_y) + q_x(v_y - p_y))^2)$. The respective collection of $\Theta(n^2)$ Voronoi surfaces fulfills Assumptions 7.1 of [12, p. 188]: (1) Each surface is an algebraic surface of maximum constant degree; (2) Each surface is totally defined (stronger than needed); and (3) Each triple of surfaces intersects in $O(1)$ points. Hence, we may apply Theorem 7.7 of [ibid., p. 191] and obtain the claimed bound. □

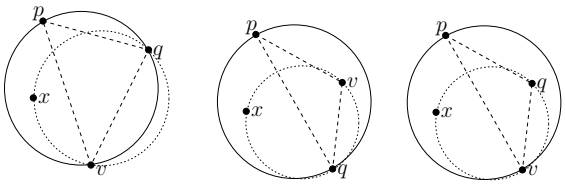
3 Containing Circle

Let $C(p, q, r)$ denote the minimum-radius circle containing three points p, q, r.

Definition 2 *Given two points p, q, the "containing-circle distance" $\mathcal{K}$ from a point v to the unordered pair (p, q) is defined as $\mathcal{K}(v, (p, q)) = \mathrm{Rad}(C(v, p, q))$.*

In our context $p \neq q$. Assume first that $v \neq p, q$. Observe that if all angles of $\triangle pqr$ are acute (or $\triangle pqr$ is right-angled), then $C(p, q, r)$ is identical to $\circ(p, q, r)$. Otherwise, if one of the angles of $\triangle pqr$ is obtuse, then $C(p, q, r)$ is the circle whose diameter is the longest edge of $\triangle pqr$. If v coincides with either p or q, $C(v, p, q)$ is the circle whose diameter is the segment pq.

Theorem 3 *Let S be a set of n points. The complexity of $V_{\mathcal{K}}^{(n)}(S)$ is $\Omega(n)$.*



(a) Acute triangle (b) Obtuse v (c) Obtuse q

Figure 1: If p, q have a non-empty region in $V_{\mathcal{K}}^{(n)}(S)$, then pq is an edge in $\mathrm{DT}(S)$

Proof. For simplicity assume that each point from S has a unique closest neighbor in S. For each $p \in S$, consider its closest neighbor q. Then, the points of pq lying sufficiently close to p belong to the region of (p, q) in $V_{\mathcal{K}}^{(n)}(S)$, which is thus non-empty. Since no region is thereby encountered more than twice, $V_{\mathcal{K}}^{(n)}(S)$ has at least $\lceil n/2 \rceil$ non-empty regions. □

Theorem 4 *Let S be a set of n points. The complexity of $V_{\mathcal{K}}^{(n)}(S)$ is $O(n^{2+\varepsilon})$ (for any $\varepsilon > 0$).*

Proof. Let a point v belong to a non-empty region of (p, q). No matter if $\triangle vpq$ is acute (Fig. 1(a)), $\triangle vpq$ is obtuse with v the obtuse vertex (Fig. 1(b)), or $\triangle vpq$ is obtuse with p or q the obtuse vertex (Fig. 1(c)), $C(v, p, q)$ cannot contain any other point $x \in S$. Otherwise, regardless of the location of x in $C(v, p, q)$, we will always have $\mathcal{K}(v, (p, q)) > \mathcal{K}(v, (x, q))$, a contradiction. This follows from the fact [6, Lemma 4.14] that given a point set K and its minimum enclosing circle C, removing from K one of the (two or three) points defining C will reduce the radius of the minimum enclosing circle. Thus, there is a circle containing p, q that is empty of any other point of S. Therefore, pq is a Delaunay edge of S. Thus, there are $O(n)$ pairs of sites in S that have non-empty regions in $V_{\mathcal{K}}^{(n)}(S)$. Furthermore, the Voronoi surface of (p, q) is made of a constant number of patches, each of which is "well-behaved" in the sense discussed above. Again, the complexity of the lower envelope of these $O(n)$ surfaces is $O(n^{2+\varepsilon})$ (for any $\varepsilon > 0$). □

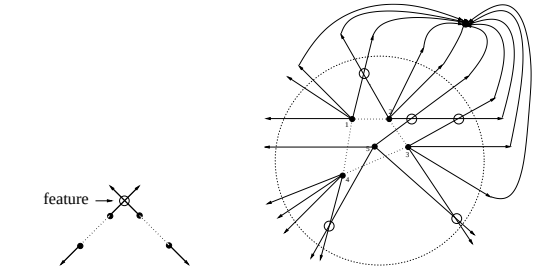
Theorem 5 *Let S be a set of n points. The complexity of $V_{\mathcal{K}}^{(f)}(S)$ is $O(n^{4+\varepsilon})$ (for any $\varepsilon > 0$).*

Proof. Again, we prove the claim via the upper envelope of $\Theta(n^2)$ "well-behaved" Voronoi surfaces. □

4 View Angle

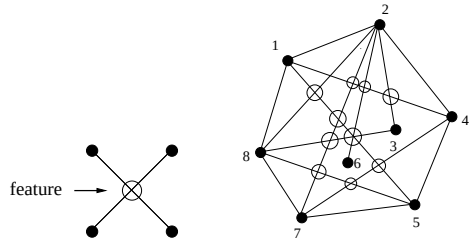
Definition 3 *Given two points p, q, the "view-angle distance" $\mathcal{V}$ from a point v to the unordered pair (p, q) is defined as $\mathcal{V}(v, (p, q)) = \measuredangle pvq$.*

Similarly to $\mathcal{C}$, the view-angle function is undefined at the given points. For fixed points p, q, the curve $\mathcal{V}(v, (p, q)) = \pi$ is the open segment pq, while the curve $\mathcal{V}(v, (p, q)) = 0$ is the line $\overline{pq}$ excluding the closed segment pq. The curve $\mathcal{V}(v, (p, q)) = \pi/2$ is the circle with diameter pq (excluding p and q).



(a) Intersection point (b) Graph

Figure 2: The graph of $V_{\mathcal{V}}^{(n)}(S)$



(a) Intersection point (b) Graph

Figure 3: The graph of $V_{\mathcal{V}}^{(f)}(S)$

Theorem 6 *Let S be a set of n points. The complexity of $V_{\mathcal{V}}^{(n)}(S)$ is $\Omega(n^4)$.*

Proof. Let S be a set of n points. Intersections of the complements of two segments defined by two pairs of points (w.r.t. the supporting lines, Fig. 2(a)) are features of $V_{\mathcal{V}}^{(n)}(S)$. Create a geometric graph G on the given points, in which each segment's complement defines two edges. Add one additional point far away from the convex hull of S, and connect it (without intersections) to all the rays (Fig. 2(b)). We can now use the crossing-number lemma for bounding the number of intersections. The lemma tells us that every drawing of a graph with n vertices and $m \geq 4n$ edges (without self or parallel edges) has $\Omega(m^3/n^2)$ crossing points [1, 10]. In our case $m = n(n-1)$, so the number of intersection points in G is $\Omega(n^4)$. □

Theorem 7 *Let S be a set of n points. The complexity of both $V_{\mathcal{V}}^{(n|f)}(S)$ is $O(n^{4+\varepsilon})$ (for any $\varepsilon > 0$).*

Proof. For analyzing $V_{\mathcal{V}}^{(n|f)}$(S) consider the function $(-\cos \measuredangle pvq)$ instead of $\measuredangle pvq$. This is allowable since $\cos(\cdot)$ is strictly decreasing in $[0, \pi]$. By the cosine law, $-\cos \measuredangle pvq = (|pq|^2 - |vp|^2 - |vq|^2)/(2|vp||vq|)$. Hence, the $\Theta(n^2)$ Voronoi surfaces fulfill Assumptions 7.1 of [12, p. 188]. As above, apply Theorem 7.7 of [ibid., p. 191] to obtain the claimed bound. □

Theorem 8 *Let S be a set of n points. The complexity of $V_{\mathcal{V}}^{(f)}(S)$ is $\Omega(n^4)$.*

Proof. Given a set S of n points, we count the intersections of pairs of line segments defined by pairs of points of S (Fig. 3(a)). Create a geometric graph on the points of S, in which edges are the line segments connecting pairs of points (Fig. 3(b)). The intersections of segments defined by all pairs of points define features of $V_{\mathcal{V}}^{(f)}(S)$, since along these segments the view-angle function assumes its maximum possible value, π. We can now use the crossing-number lemma for counting these intersections. The graph with n vertices and $m \geq 4n$ edges (without self or parallel edges) has $\Omega(m^3/n^2)$ crossing points [1, 10]. In this case $m = n(n-1)/2$, hence the claimed bound. □

Results by Asano et al. [2] imply that the edges of $V_{\mathcal{V}}^{(n|f)}(S)$ are pieces of polynomial curves of degree at most 3. However, the structure of the part of $V_{\mathcal{V}}^{(f)}(S)$ that lies outside the convex hull $\mathcal{CH}(S)$ of S is fairly simple: it is given by the arrangement of lines supporting the edges of $\mathcal{CH}(S)$. This arrangement can be computed by a standard incremental algorithm in optimal $\Theta(k^2)$ time and space, where k is the number of vertices of $\mathcal{CH}(S)$. Each cell of the arrangement should then be labeled with a pair of sites from S, to the Voronoi region of which it belongs; this extra task can be completed within the same complexity bounds.

5 Parameterized Perimeter

Definition 4 *Given two points p, q and a real constant $c \geq -1$, the "parameterized perimeter distance" $\mathcal{P}_c$ from a point v to the unordered pair (p, q) is defined as $\mathcal{P}_c(v, (p, q)) = |vp| + |vq| + c \cdot |pq|$.*

We require that $c \geq -1$, since allowing $c < -1$ would result in negative distances. Letting $c = -1$ results in a distance function that equals 0 for all the points on the segment pq. If $c = 0$, we deal with a "sum of distances" distance function introduced in [5] and recently revisited in [13]. For $c = 1$, the above definition yields the "perimeter" distance function $\mathcal{P}(v, (p, q)) = \mathrm{Per}(\triangle vpq)$.

It was proven [8] that the complexity of the 2-site perimeter Voronoi diagram of n points is $O(n^{2+\varepsilon})$. The key observation was that any pair of sites that has a non-empty region in the perimeter diagram also has a non-empty region in the sum-of-distances diagram. Consequently, the number of such pairs is linear in n. Again, one applies the Davenport-Schinzel machinery and conclude the claimed upper bound on the complexity of the diagram. We provide here an alternative and much simpler proof of the same bound, which generalizes to the case of "parameterized perimeter" distance function for any $c \geq 0$.

Theorem 9 *Let S be a set of n points. The complexity of $V_{\mathcal{P}}^{(n)}(S)$ is $O(n^{2+\varepsilon})$ (for any $\varepsilon > 0$).*

Proof. Refer to Fig. 4. Let $p, q \in S$ be two sites which have a non-empty region in $V_{\mathcal{P}}^{(n)}(S)$, and let v be a point in this region. In addition, let ℓ be the perpendicular bisector of the segment pq. Assume, w.l.o.g., that $|vp| \leq |vq|$.

Consider the ellipse O_{vpq} passing through q with v and p as foci. By definition, for any point s inside this

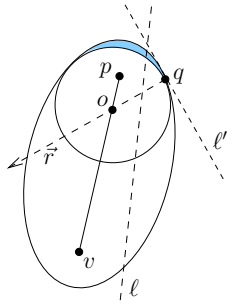


Figure 4: An empty circle containing sites in $\mathcal{P}$

ellipse we have $|vs| + |ps| < |vq| + |pq|$. Therefore,

$$\begin{aligned} \mathcal{P}(v,(p,s)) &= |vs| + |ps| + |vp| \qquad (1) \\ &< |vq| + |pq| + |vp| = \mathcal{P}(v,(p,q)). \end{aligned}$$

Hence, $s \notin S$, otherwise v would belong to the region of (p,s) instead of that of (p,q). Therefore, O_{vpq} is empty of any sites other than p and q.

Now consider the line ℓ' that is tangent to O_{vpq} at q, and the ray $\vec{r}$ perpendicular to ℓ' at q and passing through O_{vpq}. It is a known property of ellipses that this ray bisects the angle $\measuredangle vqp$, thus, it intersects the segment vp, say, at point o. The circle C centered at o and passing through q is tangent to O_{vpq} at q (as well as at another point), and is entirely contained in O_{vpq}. Since $|vp| \le |vq|$, it follows that C also contains p. (If p were on the extension of vp in the shaded area, a contradiction would easily be obtained by using the triangle inequality: $|op| > |oq|$, hence $|vp| = |ov| + |op| > |ov| + |oq| > |vq|$, a contradiction to $|vp| \le |vq|$.) Since O_{vpq} is empty of sites (except p, q), so is C. Therefore, pq is an edge of the Delaunay triangulation of S. The number of such edges is linear in n. Hence, there are $\Theta(n)$ respective surfaces of these pairs of sites. One can now apply the standard Davenport-Schinzel machinery, and the claim follows. □

Finally, we state the following theorem.

Theorem 10 *Let S be a set of n points.*
(a) The complexity of $V^{(n)}_{\mathcal{P}_{-1}}(S)$ is $\Omega(n^4)$ and $O(n^{4+\varepsilon})$ (for any $\varepsilon > 0$).
(b) If there is a unique closest pair $p, q \in S$, then when $c \to \infty$, the complexity of $V^{(n)}_{\mathcal{P}_c}(S)$ is asymptotically 1.
(c) For $c \ge 0$, the complexity of $V^{(n)}_{\mathcal{P}_c}(S)$ is $O(n^{2+\varepsilon})$ (for any $\varepsilon > 0$).

The easy proofs of Theorems 10(a,b), as well as the proof of Theorem 10(c) (which is a nontrivial generalization of the proof of the special case $c = 1$), are provided in the full version of the paper.

6 Conclusion

In this paper, we have investigated 2-site Voronoi diagrams of point sets with respect to a few geometric distance functions. The Voronoi structures obtained in this way cannot be explained in terms of the previously known kinds of Voronoi diagrams (which is the case for the 2-site distance functions thoroughly analyzed in [5]), what makes them particularly interesting. On the other hand, our results can be exploited to advance research on Voronoi diagram for segments. Potential directions for future work include consideration of other distance functions, and generalizations to higher dimensions and to k-site Voronoi diagrams.

Acknowledgments

Work on this paper by the first author was performed while he was visiting Tufts University, Medford, MA. Work of the last author was partially supported by Russian Foundation for Basic Research (grant 10-07-00156-a).

References

[1] M. Ajtai, V. Chvátal, M. Newborn, and E. Szemerédi, Crossing-free subgraphs, *Annals of Discrete Mathematics*, 12 (1982), 9–12.

[2] T. Asano, N. Katoh, H. Tamaki, and T. Tokuyama, Angular Voronoi diagram with applications, *Proc. 3rd Int. Symp. on Voronoi Diagrams in Science and Engineering*, Banff, Canada, 32–39, 2006.

[3] T. Asano, N. Katoh, H. Tamaki, and T. Tokuyama, Voronoi diagrams with respect to criteria on vision information, *Proc. 4th Int. Symp. on Voronoi Diagrams in Science and Engineering*, Pontypridd, Wales, UK, 25–32, 2007.

[4] F. Aurenhammer, Voronoi diagram—A survey of a fundamental geometric data structure, *ACM Computing Surveys*, 23 (1991), 345–405.

[5] G. Barequet, M.T. Dickerson, and R.L.S. Drysdale, 2-point site Voronoi diagrams, *Discrete Applied Mathematics*, 122 (2002), 37–54.

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry, Algorithms, and Applications* (3rd ed.), Springer-Verlag, Berlin, 2008.

[7] M.T. Dickerson and D. Eppstein, Animating a continuous family of two-site Voronoi diagrams (and a proof of a bound on the number of regions), *Proc. 25th ACM Symp. on Computational Geometry*, Aarhus, Denmark, 92–93, 2009.

[8] I. Hanniel and G. Barequet, On the triangle-perimeter two-site Voronoi diagram, *Proc. 6th Int. Symp. on Voronoi Diagrams*, Copenhagen, Denmark, 129–136, 2009.

[9] D. Hodorkovsky, *2-Point Site Voronoi Diagrams*, M.Sc. Thesis, The Technion—Israel Inst. of Technology, Haifa, Israel, 2005.

[10] F.T. Leighton, Complexity Issues in VLSI, MIT Press, Cambridge, MA, 1983.

[11] A. Okabe, A. Boots, B. Sugihara, and S.N. Chui, Spatial Tesselations, 2nd ed., Wiley, 2000.

[12] M. Sharir and P.K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Application*, Cambridge University Press, 1995.

[13] K. Vyatkina and G. Barequet, On 2-site Voronoi diagrams under arithmetic combinations of point-to-point distances, *Proc. 7th Int. Symp. on Voronoi Diagrams*, Québec City, Québec, Canada, 33–41, 2010.

Covering and Piercing Disks with Two Centers*

Hee-Kap Ahn† Christian Knauer‡ Sang-Sub Kim† Lena Schlipf§ Hyeon-Suk Na¶
Chan-Su Shin‖ Antoine Vigneron **

Abstract

We consider some variations of the two center problem. Let $\mathcal{D}$ denote a set of disks in the plane. We first study the problem of finding two smallest congruent disks such that each disk in $\mathcal{D}$ is intersected by one of these disks. Then we study the problem of covering the set $\mathcal{D}$ by two smallest congruent disks.

1 Introduction

There has been a fair amount of works on *the two center problem* for points: Given a set P of n points in the plane, find two smallest congruent disks that cover all points in P. So far, the best known algorithms run in $O(n \log^2 n \log^2 \log n)$ worst-case time [1] and in $O(n \log^2 n)$ expected time [4].

In this paper we consider two versions of the problem where the input consists of disks instead of points, so called *intersecting* and *covering* problems: Given a set $\mathcal{D}$ of n disks, compute two smallest congruent disks C_1 and C_2 such that each $D \in \mathcal{D}$ is intersected by C_1 or C_2 for the intersecting problem or is contained by the union of C_1 and C_2 for the covering problem.

The intersecting problem can be also formulated as a piercing problem: Compute the smallest value δ such that increasing the radius of every disk in $\mathcal{D}$ by δ makes the set pierceable by two points, meaning that there exist two points such that each disk contains at least one of the points. We present a simple $O(n^3)$ algorithm for this problem and algorithms which run in $O(n^2 \log^3 n)$ expected time and $O(n^2 \log^4 n \log \log n)$ time in the worst case.

The covering problems has also two different cases: In the *restricted case* each disk $D \in \mathcal{D}$ has to be fully covered by one of the disks C_1 or C_2. In the *general case* a disk $D \in \mathcal{D}$ can be covered by the union of C_1 and C_2. We show how the algorithms for the intersecting problem can be used to solve the restricted covering case. We complement these results by giving efficient approximation algorithms for the restricted and the general covering cases.

2 Preliminaries

The radius of a disks D is denoted by $r(D)$ and its center by $c(D)$. The distance between two disks D_1 and D_2 is denoted by $d(D_1, D_2)$ and is defined as $d(D_1, D_2) = d(c(D_1), c(D_2)) + r(D_1) + r(D_2)$.

3 Intersecting Disks with Two Disks

Given a set of disks $\mathcal{D} = \{D_1, \ldots, D_n\}$, we want to find two smallest congruent disks C_1 and C_2 such that for every disk $D_i \in \mathcal{D}$, D_i has a nonempty intersection with C_1 or C_2. Based on the observation below, we can design an algorithm that finds the optimal solution in $O(n^3)$ time.

Observation 1 *Let ℓ be the bisector of an optimal solution C_1 and C_2. Then, $C_i \cap D \neq \emptyset$ for any $D \in \mathcal{D}$ whose center lies in the same side of the center of C_i, for $i = \{1, 2\}$.*

For every bipartition of the centers of the disks in $\mathcal{D}$, we solve the 1-center problem for the disks in each partition. Then the center with larger radius is the solution for the 2-center problem restricted to the bipartition. We return the best one over all bipartitions. Since there are $O(n^2)$ such bipartitions and each 1-center problem can be solved in linear time [7], this algorithm runs in time $O(n^3)$.

To improve the time complexity, we formulate the problem in a different way as follows. For a real number δ, a *δ-inflated* disk of a disk D_i, denoted by $D_i(\delta)$,

*Work by Ahn and Kim was supported by the National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technology Development. Work by Schlipf was supported by the German Science Foundation (DFG) within the research training group 'Methods for Discrete Structures'(GRK 1408).

†Department of Computer Science and Engineering, POSTECH, Pohang, Korea. {heekap, helmet1981}@postech.ac.kr

‡Institute of Computer Science, Universität Bayreuth, 95440 Bayreuth, Germany. christian.knauer@uni-bayreuth.de

§Institute of Computer Science, Freie Universität Berlin, 14195 Berlin, Germany. schlipf@mi.fu-berlin.de

¶School of Computing, Soongsil University, Seoul, Korea. hsnaa@ssu.ac.kr

‖Department of Digital and Information Engineering, Hankuk University of Foreign Studies, Yongin, Korea. cssin@hufs.ac.kr

**Division of Mathematical and Computer Sciences and Engineering, KAUST, Thuwal, Saudi Arabia. antoine.vigneron@kaust.edu.sa

is a disk concentric to D_i with radius $r(D_i)+\delta$. Consider the following decision problem:

> Given a value δ, do there exist two points, p_1 and p_2, in the plane such that $D_i(\delta) \cap \{p_1, p_2\} \neq \emptyset$ for every $D_i \in \mathcal{D}$.

We let δ^* be the minimum value for which the decision problem answers "yes". To ensure that the radius of δ-*inflated* disks is nonnegative, we require that $\delta \geq -r_{\min}$, where $r_{\min}$ be the minimum of radii of all disks in $\mathcal{D}$. Without loss of generality, we also assume that no disk contains another disk in its closure, and that at most three δ-inflated disks intersect at a point (on their boundaries) for all disks in $\mathcal{D}$.

3.1 A decision algorithm

Given a value δ, we construct the arrangement of n δ-inflated disks in the plane. The arrangement consists of $O(n^2)$ cells. We traverse cells in the arrangement in depth-first manner and do the followings: We place one center point, say p_1, in a cell. The algorithm returns "yes" if all the disks that do not contain p_1 have a nonempty common intersection. Otherwise, we move p_1 to a neighboring cell, and repeat the test until we visit every cell. To test all cells in a naive way leads to a running time of $O(n^3)$.

To do the test efficiently, we partition the disks into $O(\sqrt{n})$ subsets with respect to their indices such that the first subset $\mathcal{D}_1$ consists of the first $\lceil\sqrt{n}\rceil$ disks $D_1, \cdots, D_{\lceil\sqrt{n}\rceil}$ and $\mathcal{D}_2$ consists of the following $\lceil\sqrt{n}\rceil$ disks, and so on. For each subset $\mathcal{D}_i$, we maintain a data structure representing the common intersection $I_i = \bigcap_{p \notin D, D \in \mathcal{D}_i} D$, of the disks in the set that do not contain p in its closure.

When we move p to a neighboring cell, at most one disk changes its status - therefore we update the common intersection of the corresponding group only, and this can be done in $O(\sqrt{n}\log n)$ time. If every I_i is nonempty, we decide whether they have a nonempty common intersection or not in $O(\sqrt{n}\log n)$ expected time using the randomized convex programming [3, 10] or in $O(\sqrt{n}\log^2 n)$ time using the deterministic convex programming [2]. Therefore the algorithm takes $O(n^{2.5}\log^2 n)$ time in total.

We can improve the time complexity further to $O(n^2 \log^2 n \log\log n)$ time or $O(n^2 \log^2 n)$ expected time by representing a traversal of cells as a set of intervals along a time line of the traversal, storing the set in segment trees, and checking the emptiness of the disk intersection in each cell with a help of the segment trees in $O(\log^2 n)$ expected time [3, 10] or $O(\log^2 n \log\log n)$ worst-case time [2]. Details will be found in the full version of the paper.

Lemma 1 *Given a value δ, we can decide in $O(n^2 \log^2 n \log\log n)$ time or in $O(n^2 \log^2 n)$ expected time whether there exist two points such that every δ-inflated disk is intersected by at least one of them.*

Note that the decision algorithm returns two solution points if the answer is "yes".

3.2 Finding δ^*

Lemma 2 *When $\delta = \delta^*$, either p_1 or p_2 is a common boundary point of three δ^*-inflated disks, a tangent point of two δ^*-inflated disks or a δ^*-inflated disk with radius zero.*

Thus, we consider only discrete values of δ for which one of the events defined in Lemma 2 occurs. If p_1 or p_2 is a δ-inflated disk with zero radius, that is, $\delta = -r_{\min}$, we can test whether the common intersection of the remaining δ-inflated disks is empty or not in $O(n)$ time. If p_1 or p_2 is a tangent point of two inflated disks, then we can collect such $O(n^2)$ tangent points (also associated radii) from all pairs of disks, and perform the binary search over the sorted radii by decision algorithm in Lemma 1 in $O(n^2 \log^3 n)$ expected time or $O(n^2 \log^3 n \log\log n)$ worst-case time. So from now on, we assume without loss of generality that p_1 is an intersection point of three δ-inflated disks.

For this case, we construct a frustum $f_i \in \mathbb{R}^3$ for each disk $D_i \in \mathcal{D}$. The bottom base of the frustum f_i is $D_i(-r_{\min})$ lying in the plane $z = -r_{\min}$. The intersection of f_i and the plane $z = \delta$ is $D_i(\delta)$. The top base of f_i is $D_i(\delta_{\max})$, where $\delta_{\max}$ is the radius of the smallest disk intersecting all disks in $\mathcal{D}$. Clearly, the optimal value of δ is in $[-r_{\min}, \delta_{\max}]$.

Let $p = (x, y)$ be the intersection point of the disks $D_i(\delta)$, $D_j(\delta)$, and $D_k(\delta)$. Then the point $p' = (x, y, \delta)$ is the intersection point of three frustums f_i, f_j, and f_k. Thus, p_1 is the projection of a point p' which is an intersection point of three frustums, i.e., a vertex of the arrangement $\mathcal{A}$ of the n frustums $f_1, \ldots f_n$; the complexity of $\mathcal{A}$ is $O(n^3)$. Note that for each candidate for p_1, the corresponding value for δ is easily obtained, namely the height of p'_1. Since $\mathcal{A}$ contains $O(n^3)$ vertices, we need to compute them implicity.

Implicit binary search We now describe how to perform the binary search over the vertices of $\mathcal{A}$ in an implicitly way:

Binary search on a coarse list of vertices. We first randomly select $O(n^2 \log n)$ vertices from $\mathcal{A}$ by picking $O(n^2 \log n)$ triples of frustums, and sort the radii associated with them in $O(n^2 \log^2 n)$ time. By a binary search with the decision algorithm in Lemma 1, we determine two consecutive radii δ_i and δ_{i+1} such that δ^* is between δ_i and δ_{i+1}. This takes $O(n^2 \log^3 n)$ time. Since the vertices were picked randomly, the strip $W[\delta_i, \delta_{i+1}]$ bounded by the two planes $z := \delta_i$ and $z := \delta_{i+1}$ contains only $O(n)$ vertices of $\mathcal{A}$ with high probability [9, Section 5].

Zooming into the interval. We compute all the k vertices in $W[\delta_i, \delta_{i+1}]$ by a standard sweep-plane algorithm in $O(k \log n + n^2 \log n)$ time as follows: First, we compute the intersection of the sweeping plane at $z := \delta_i$ with the frustums $f_1, \dots f_n$. This intersection forms a two-dimensional arrangement of $O(n)$ circles with $O(n^2)$ total complexity, and we can compute it in $O(n^2 \log n)$ time. We next construct the portion of the arrangement $\mathcal{A}$ in $W[\delta_i, \delta_{i+1}]$ incrementally by sweeping a vertical plane from the intersection at $z := \delta_i$ towards $z := \delta_{i+1}$. As a result, we can compute the $k = O(n)$ vertices (and the corresponding $O(n)$ radii) in $W[\delta_i, \delta_{i+1}]$ in $O(n \log n)$ time. We abort the sweep if the number k of vertices inside the strip becomes too large and restart the algorithm with a new random sample. This happens only with small probability. In order to find the minimum value δ^*, we perform a binary search on these $O(n)$ radii we just computed, using the decision algorithm in Lemma 1. This takes $O(n^2 \log^3 n)$ expected time. The solution pair of points p_1 and p_2 can also be found by the decision algorithm.

To get a deterministic algorithm, we use the parametric search technique with deterministic decision algorithm in Lemma 1, but the running time increases by one $(\log n)$-factor, so the total time becomes $O(n^2 \log^4 n \log\log n)$ time. Details can be found in the full version of this paper.

Theorem 3 *Given a set $\mathcal{D}$ of n disks in the plane, we can compute two smallest congruent disks whose union intersects every disk in $\mathcal{D}$ in $O(n^2 \log^3 n)$ expected time or in $O(n^2 \log^4 n \log\log n)$ worst-case time.*

4 Covering Disks with Two Disks

Given a set of disks $\mathcal{D} = \{D_1, \dots, D_n\}$ in the plane, we want to find two smallest congruent disks C_1 and C_2, such that every disk $D_i \in \mathcal{D}$ is covered by C_1 or C_2. As mentioned in the introduction, we distinguish between two cases: The *restricted case* where each disk D_i has to be fully covered by C_1 or C_2 and the *general case* where a disk D_i can be covered by $C_1 \cup C_2$.

4.1 The Restricted Case

Observation 2 *Let ℓ be the bisector of an optimal solution C_1 and C_2. Then, $D \subset C_i$ for every $D \in \mathcal{D}$ whose center lies in the same side of the center of C_i, for $i = \{1, 2\}$.*

Hence, the restricted covering case can be solved in $O(n^3)$ time, since the smallest disk covering a set of disks can be computed in linear time [8] and there are $O(n^2)$ bipartitions of the centers of the disks.

The algorithm from Section 3 can be also be used to solve this problem. For this we consider the decision problem: Given a set of n disks $\mathcal{D}$ and a value δ, do there exist two disks C_1, C_2 with radius δ, such that each disk $D_i \in \mathcal{D}$ is covered by either C_1 or C_2. This implies that for each disk $D_j \in \mathcal{D}$ covered by C_i holds: $d(c(D_j), c(C_i)) + r(D_j) \leq \delta$, for $i = \{1, 2\}$. It clearly holds that $\delta \geq r_{\max}$, where $r_{\max}$ is the maximum of radii of all disks in $\mathcal{D}$. We can formulated the problem in a different way.

> Given a value δ, do there exist two points, p_1 and p_2, such that $D_i^*(\delta) \cap \{p_1, p_2\} \neq \emptyset$ for every $D_i \in \mathcal{D}$, where $D_i^*(\delta)$ is a disk concentric to D_i and whose radius is $\delta - r(D_i) \geq 0$.

Since $\delta \geq r_{\max}$, we add an initialization step, in which every disk D_i is replaced by a disk concentric to D_i with radius $r_{\max} - r(D_i)$. Then we can use the algorithm from Section 3 in order to solve the restricted covering problem.

Theorem 4 *Given a set of n disks $\mathcal{D}$ in the plane, we can compute two smallest congruent disks such that each disk in $\mathcal{D}$ is covered by one of the disks in $O(n^2 \log^3 n)$ expected time or in $O(n^2 \log^4 n \log\log n)$ worst-case time.*

Constant factor approximation Algorithm 1 computes a $2/\sqrt{3}$-approximation in $O(n \log n)$ time. OneCover($\mathcal{U}$), which is used as a subroutine, computes the smallest disk covering a set of disks $\mathcal{U}$. Algorithm 1 runs in $O(n \log n)$ time, since the diameter of $\mathcal{D}$ can be computed in $O(n \log n)$ [7] and OneCover($\mathcal{U}$) in linear time [8]. The approximation factor of $2/\sqrt{3}$ can be proven by making a case distinction whether D_1 and D_2 are covered by the same disk in the optimal solution or by different disks, and by using Jung's theorem [5].

Algorithm 1

```
1: U1 = ∅ and U2 = ∅.
2: Compute the diametral pair D1 and D2, that is,
   d(D1, D2) is the diameter of D.
3: U1 = U1 ∪ {D1} and U2 = U2 ∪ {D2}.
4: for all Di ∈ D do
5:     if d(Di, D1) < d(Di, D2) then
6:         U1 = U1 ∪ {Di}
7:     else
8:         U2 = U2 ∪ {Di}
9: Compute C1 =OneCover(U1) and C2 =OneCover(U2)
10: return C1 and C2
```

At the expense of increasing the approximation factor by a factor of $\sqrt{2}$, we can improve the running time to $O(n)$ by replacing the computation of the diameter of $\mathcal{D}$ in Algorithm 1 by computing a $1/\sqrt{2}$-approximation of the diameter in $O(n)$ time.

Theorem 5 *Given a set of n disks $\mathcal{D}$. For the restricted covering case a $2/\sqrt{3}$-approximation can be computed in $O(n \log n)$ time and a $2\sqrt{2}/\sqrt{3}$-approximation in $O(n)$ time.*

$(1+\epsilon)$-**approximation** Recall Observation 2. We show how to compute an optimal solution in $O(n \log n)$ time if the orientation of the bisector is given and explain how this algorithm is used in order to obtain a $(1+\epsilon)$ approximation.

Fixed Orientation. W.l.o.g, assume that the bisector is vertical. After sorting the centers of all $D_i \in \mathcal{D}$ by their x-values, we sweep a vertical line ℓ from left to right, and maintain two sets $\mathcal{D}_1$ and $\mathcal{D}_2$: $\mathcal{D}_1$ contains all disks whose centers lie to the left of ℓ and $\mathcal{D}_2 = \mathcal{D} \setminus \mathcal{D}_1$. Let C_1 be the smallest disk covering $\mathcal{D}_1$ and C_2 the smallest disk covering $\mathcal{D}_2$. While sweeping ℓ from left to right, the radius of C_1 is nondecreasing and the radius of C_2 nonincreasing and we want to compute $\min \max(r(C_1), r(C_2))$. Hence, we can perform a binary search on the list of all centers. Each step takes $O(n)$ time, thus we achieve a total running time of $O(n \log n)$.

Sampling. We use $2\pi/\epsilon$ sample orientations chosen regularly over 2π, and compute for each orientation the solution in $O(n \log n)$ time. The approximation factor can be proven by showing that there is a sample orientation that makes angle at most ϵ with the optimal bisector. The solution for this line is at most $(1+\epsilon)$ times the optimal solution.

Theorem 6 *Given a set $\mathcal{D}$ of n disks in the plane, a $(1+\epsilon)$ approximation for the restricted covering problem for $\mathcal{D}$ can be computed in $O(n \log n/\epsilon)$ time.*

4.2 The General Case

Lemma 7 *A solution for the restricted case is a $(\sqrt{2}+1)/2$-approximation for the general case.*

Theorem 8 *Given a set $\mathcal{D}$ of n disks, a $(\sqrt{2}+1)/\sqrt{3}$-approximation for the general covering problem for $\mathcal{D}$ can be computed in $O(n \log n)$ time and a $(2+\sqrt{2})/\sqrt{3}$-approximation in $O(n)$ time.*

$(1+\epsilon)$-**Approximation** First, we compute a $(2+\sqrt{2})/\sqrt{3}$-approximation for the general covering case in $O(n)$ time as explained before. Let C_1^{apx}, C_2^{apx} be the solution disks. We can assume that $d(c(C_1^{\text{apx}}), c(C_2^{\text{apx}})) \leq (2+2(2+\sqrt{2})/\sqrt{3}) r(C_1^{\text{apx}}) < 6r(C_1^{\text{apx}})$, otherwise C_1^{apx}, C_2^{apx} are already an optimal solution. We compute a smallest bounding box B of $C_1^{\text{apx}} \cup C_2^{\text{apx}}$ and an $O(1/\epsilon) \times O(1/\epsilon)$ grid on B. The length of each grid cell is $< 6\epsilon r(C_1^{\text{apx}}) < 12\epsilon r^*$, where r^* is the radius of the optimal solution disks. Each disk is replaced in $O(n/\epsilon)$ time by the grid points which are closest to its boundary and lie inside the disk. If a disk has no grid point lying inside, it is replaced by the grid point which is closest to this disk. In total we have a set of $O(1/\epsilon^2)$ points for which we solve the two center problem, meaning we compute the smallest two disks E_1, E_2 that cover this point set in $O(\frac{1}{\epsilon^2} \log^2 \frac{1}{\epsilon} \log^2 \log \frac{1}{\epsilon})$ time [1]. It holds that $r(E_1) = r(E_2) \leq r^*$ and if the radii of E_1, E_2 are increased by the length of the diagonal of a grid cell, which is $< 6\epsilon\sqrt{2} r(C_1^{\text{apx}})$, these disks cover $\mathcal{D}$ and their radii are $< r^*(1+12\sqrt{2}\epsilon)$. The total running time is $O(\frac{1}{\epsilon^2} \log^2 \frac{1}{\epsilon} \log^2 \log \frac{1}{\epsilon} + \frac{n}{\epsilon})$.

In order to improve the running time for very small ϵ, i.e., $\epsilon \leq 1/\log n$, we compute the set of grid points from the union of the disks in $\mathcal{D}$. The union can be computed in $O(n \log n)$ time and its complexity is $O(n)$ [6]. Hence, the replacement of the disks by grid points takes $O(n \log n + 1/\epsilon^2)$ time.

Theorem 9 *Given a set $\mathcal{D}$ of n disks in the plane, a $(1+\epsilon)$-approximation for $\mathcal{D}$ in the general covering case can be computed in $O(\frac{1}{\epsilon^2} \log^2 \frac{1}{\epsilon} \log^2 \log \frac{1}{\epsilon} + \min\{\frac{n}{\epsilon}, n \log n\})$ time.*

References

[1] T. M. Chan. More planar two-center algorithms. *Comput. Geom. Theory Appl.*, 13:189–198, 1997.

[2] T. M. Chan. Deterministic algorithms for 2-d convex programming and 3-d online linear programming. *J. Algorithms*, 27(1):147–166, 1998.

[3] K. L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42:488–499, 1995.

[4] D. Eppstein. Faster construction of planar two-centers. In *Proc. SODA'97*, pages 131–138. ACM and SIAM, 1997.

[5] H. Jung. Über den kleinsten Kreis, der eine ebene Figur einschließt. *J. Reine Angew. Math.*, 137:310–313, 1910.

[6] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.

[7] M. Löffler and M. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Comput. Geom. Theory Appl.*, 43:419–433, 2010.

[8] N. Megiddo. On the ball spanned by balls. *Discrete Comput. Geom.*, 4:605–610, 1989.

[9] K. Mulmuley. *Computational Geometry - An Introduction Through Randomized Algorithms.* Prentice Hall, 1994.

[10] M. Sharir and E. Welzl. A Combinatorial Bound for Linear Programming and Related Problems. In *Proc. STACS'92*, pages 569–579. Springer-Verlag, 1992.

The L_∞ Hausdorff Voronoi diagram revisited

Evanthia Papadopoulou* Jinhui Xu†

Abstract

We revisit the L_∞ Hausdorff Voronoi diagram of clusters of points, equivalently, the L_∞ Hausdorff Voronoi diagram of rectangles, and present a plane sweep algorithm for its construction, generalizing and improving upon previous results. We show that its structural complexity is $\Theta(n+m)$, where n is the number of given clusters and m is the number of *essential* pairs of *crossing* clusters. The algorithm runs in $O((n+M)\log n)$ time and $O(n+M)$ space, after an $O((n+V)\log n)$-time preprocessing step, where M, V reflect special crossings that are *potentially essential*; m, M, V are $O(n^2)$, $m \leqslant M$, but $m = M, m \geqslant V$, in the worst case. In practice, $m, M, V << n^2$, as the total number of crossings in the motivating application is typically small, even compared to n. For non-crossing rectangles the algorithm runs in optimal $O(n \log n)$-time and $O(n)$-space.

1 Introduction

Given a set S of point clusters in the plane, the *Hausdorff Voronoi diagram* of S, denoted *HVD(S)*, is a subdivision of the plane into regions such that the *Hausdorff Voronoi region* of a cluster P, denoted *hreg(P)*, is the locus of points *closer* to P than to any other cluster in S, where distance between a point t and a cluster P is measured as the *farthest distance* between t and any point in P, $d_f(t,P) = \max\{d(t,p), p \in P\}$.

$$hreg(P) = \{x \mid d_f(x,P) < d_f(x,Q), \forall Q \in S\}$$

hreg(P) is subdivided into finer regions by the *farthest Voronoi diagram* of P, *FVD(P)*. The farthest distance $d_f(t,P)$ is equivalent to the *Hausdorff distance*[1] between t and P. In the L_∞ metric[2], $d_f(t,P)$ is equivalent to $d_f(t,P')$, where P' is the minimum enclosing axis-aligned rectangle of P. Thus, the L_∞ Hausdorff Voronoi diagram of S is equivalent to the L_∞ Hausdorff Voronoi diagram of the set S' of the minimum enclosing rectangles of all clusters in S. In this abstract the terms cluster and rectangle are used interchangeably. Once the cluster minimum enclosing rectangles are available, individual cluster points have no further influence.

The Hausdorff Voronoi diagram has appeared in the literature under different names having been motivated by different problems [6, 1, 9, 13, 10, 5, 15]. It first appeared in [6] as the *cluster Voronoi diagram*, where several combinatorial bounds were derived. A tight combinatorial bound on its structural complexity in the Euclidean plane was given in [10]. The L_∞ version of the problem was introduced in [9] as a solution to the VLSI *critical area extraction* problem for a defect mechanism on *contact layers*, called a *via-block*. A related, reverse type, of Voronoi diagram has been considered in [14, 2, 3].

In this paper we revisit the Hausdorff Voronoi diagram in the L_∞ metric. We provide a tight bound on its structural complexity and a plane sweep algorithm for its construction, generalizing and improving upon [9]. In particular, we show that the structural complexity of the L_∞ Hausdorff Voronoi diagram is $\Theta(n+m)$, where n is the number of input clusters and m is the number of *essential pairs* of *crossing* clusters (see Def. 1). The algorithm consists of an $O((n+V)\log n)$-time preprocessing step, followed by the main algorithm that runs in $O((n+M)\log n)$-time and $O(n+M)$-space, where M, V reflect numbers of special crossings that are *potentially essential* (see Def. 2); m, M, V are $O(n^2)$, $m \leqslant M$, but $m = M, m \geqslant V$, in the worst case. In practice, typically, $m, M, V << n^2$, as the total number of possible crossings in our application is small, even compared to n. For non-crossing rectangles the algorithm simplifies to optimal $O(n \log n)$-time and $O(n)$-space; no previous algorithm achieves an optimal bound in case of non-crossing clusters. An output sensitive version of the plane sweep construction, at the expense, however, of advanced data structures that require increased space consumption, is given in [15].

The L_∞ Hausdorff Voronoi diagram finds applications in VLSI design for manufacturability as explained in [9, 11] and it has been integrated in [16]. Due to its simplicity and the absence, in L_∞, of numerical issues, the approach is very well suited for use in applications.

*Faculty of Informatics, Università della Svizzera italiana, evanthia.papadopoulou@usi.ch. Supported in part by SNSF project 200021_127137.

†Department of Computer Science and Engineering, State University of New York at Buffalo, {jinhui}@buffalo.edu

[1] The (directed) Hausdorff distance from a set A to a set B is $h(A,B) = \max_{a\in A}\min_{b\in B}\{d(a,b)\}$. The (undirected) Hausdorff distance between A and B is $d_h(A,B) = \max\{h(A,B), h(B,A)\}$.

[2] The L_∞ distance between two points p, q is $d(p,q) = \max\{|x_p - x_q|, |y_x - y_p|\}$.

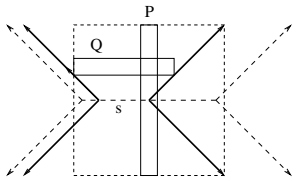


Figure 1: The L_∞ Hausdorff bisector of crossing rectangles.

2 Structural complexity and Definitions

Let S be a set of n rectangles, or a set of n point clusters in the plane, where each cluster has been substituted by its minimum enclosing rectangle. A pair of rectangles (P, Q) is called *crossing* if P and Q intersect in the shape of a cross. Given a crossing pair (P, Q), P is assumed to be at least as long as Q. For a rectangle P, let P^n, P^s, P^e, and P^w denote the north, south, east, and west edge of P respectively. P is called a horizontal (resp. vertical) if P^n is longer (resp. shorter) then P^e. The axis parallel line through edge P^i, $i = n, s, e, w$, is denoted as $l(P^i)$. The term P^i is also used to denote the main coordinate of edge P^i. The *core segment* of P is the locus of centers of all minimum enclosing squares of P, and it is given by the axis-parallel line segment of the L_∞ *farthest Voronoi diagram* of P. It can be viewed as an ordinary line segment s additively weighted with $w(s) = d_f(s, P)$. In Fig. 1, *FVD(P)* is illustrated in dashed lines and the core segment is indicated by s.

The Hausdorff bisector between two clusters P, Q is $b_h(P, Q) = \{y \mid d_f(y, P) = d_f(y, Q)\}$ and it is a subgraph of $FVD(P \cup Q)$ [13]. For a rectangle Q strictly enclosed in the interior of a minimum enclosing square of P, $b_h(P, Q)$ consists of either one (if P and Q are non-crossing) or two (if P and Q are crossing) chains, each one forming a V-shape out of the ± 1-slope rays of $FVD(P \cup Q)$; the apex of each chain is called a *V-vertex*. A V-vertex v is incident to the core segment of P and its $90°$ angle faces the portion of the plane closer to P. It is characterized as *up*, *down, right*, or *left*, depending on whether its $90°$ angle is facing north, south, east, or west respectively. In addition, it is characterized as *crossing*, if Q is crossing P, and *non-crossing*, otherwise. The minimum enclosing square of P centered at V-vertex v is also enclosing Q and it is denoted as $square(P, v)$. It is also denoted as $square(P, Q^i)$, where Q^i, is the non-crossing edge of Q that delimits one of its edges. Fig. 1 illustrates $b_h(P, Q)$ consisting of two crossing V-vertices, one right and one left; $square(P, Q^w)$ is illustrated dashed. $square(P, Q^i)$ is referred to as an *extremal minimum enclosing square* of P and Q. The V-vertices of *HVD(S)* are referred to as *Voronoi V-vertices*.

Definition 1 *A pair of crossing rectangles (P, Q) is called* essential *if there is an extremal minimum enclosing square of P and Q, $square(P, Q^i)$, that is empty of any other rectangle.*

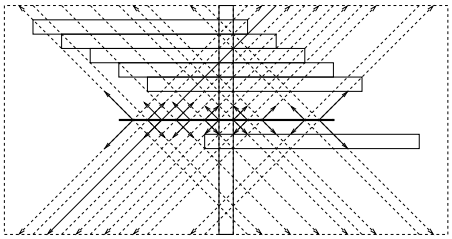


Figure 2: Collection of essential crossings.

Lemma 1 *A pair of crossing rectangles (P, Q) induces a Voronoi V-vertex in HVD(S) if and only if (P, Q) is an essential crossing.*

Combining Lemma 1 with the structural complexity results of [13] we derive the following bound.

Theorem 2 *The structural complexity of the L_∞ Hausdorff Voronoi diagram of a set S of n point clusters, equivalently n rectangles, is $\Theta(n + m)$, where m is the total number of essential crossings.*

Definition 2 *A collection of crossings for a vertical rectangle P, (P, Q_i), $i = 1, \ldots, k$, is called* a staircase, *if $Q_i^w < Q_{i+1}^w$ and $Q_i^e < Q_{i+1}^e$, $i = 1, \ldots, k$. If in addition, $square(P, Q_i^w)$ is empty of $Q_j \neq Q_i$, the staircase and its crossings are called* potentially essential. *The maximum size of a potentially essential staircase for P is the number of potentially essential crossings for P. Let M denote the total number of potentially essential crossings for all vertical rectangles in S, plus the number of essential crossings for all horizontal rectangles. Let $V = \sum V(P)\ \forall$ vertical $P \in S$, where $V(P)$ is the maximum number of pairwise non-crossing vertical rectangles that cross P.*

Fig.2 shows a collection of potentially essential crossings that are essential i.e., they all induce Voronoi V-vertices.

3 A refined plane sweep construction

In this section we revisit the plane sweep construction of the L_∞ Hausdorff Voronoi diagram [9], generalize it to crossing rectangles, and improve its time complexity to optimal in the non-crossing case. It is based on the standard plane sweep paradigm for Voronoi diagrams [7, 4] and its adaptation for line segments in L_∞ [12]. Assume a vertical sweep-line l_t sweeping the entire plane from left to right. At any instant t of the sweeping process we compute $HVD(S_t \cup l_t)$, for $S_t = \{P \in S \mid l(P^e) < t\}$. The boundary of the Voronoi region of l_t is the *wavefront* at time t. Voronoi edges and core segments incident to the wavefront are called *spike bisectors* and

spike core segments respectively. The combinatorial structure of the wavefront changes at specific *events* organized in a priority queue. We have four types of *site events*: *start-vertical-rectangle*, *end-vertical-rectangle*, *V-vertex* events (for brevity *V-events*), and *horizontal-rectangle* events. Site events are partially similar to those for ordinary line segments [12] enhanced with additional functions to handle V-vertices and disconnected Voronoi regions. *Spike events* are caused by the intersection of incident spike bisectors, and they remain the same as in the ordinary plane sweep paradigm.

The *wave-curve* of a rectangle R is the bisector between R and the sweep line l_t, at time t, $b(R, l_t) = \{y \mid d_f(y, R) = d(y, l_t)\}$, where $d(y, l_t)$ is the ordinary distance between y and l_t. In L_∞, it consists of two or three *waves*: a ray of slope -1, corresponding to $b(R^s, l_t)$, a ray of slope $+1$, corresponding to $b(R^n, l_t)$, and possibly a vertical line segment corresponding to $b(R^w, l_t)$, if appropriate. The *wavefront*, at time t, is the lower envelope, with respect to the sweep line, of the *wave-curves* of all rectangles in S_t. In L_∞, the wavefront is monotone with respect to any line of slope ± 1.

The wavefront is typically maintained as a height balanced binary tree, $\mathcal{T}$, ordered from bottom to top. Leaf nodes correspond to waves, while internal nodes correspond to spike bisectors and spike core segments revealing *breakpoints* between incident waves. In order to answer queries regarding V-vertices efficiently, we augment $\mathcal{T}$ with additional information. Each node x is augmented with a *w-max* value representing the rightmost west edge of all rectangles contributing a wave to the portion of the wavefront rooted at x, and an *x-min* value representing the minimum x-coordinate of the portion of the wavefront rooted at x. In particular, for a leaf node representing a wave of rectangle R, the *w-max* value is R^w and the *x-min* value is $+\infty$. For an internal node x, *w-max* is the maximum between the *w-max* values of its children; the *x-min* value points to the breakpoint of minimum x-coordinate among its own breakpoint and the *x-min* values of its children. These values remain the same unless a combinatorial change in the wavefront (i.e., an event) takes place.

Any Voronoi point in *HVD(S)* enters the wavefront at the time of its *priority*. The *priority* of a point v is the rightmost x-coordinate of the smallest square centered at v that is entirely enclosing the rectangle P that induces v. The priority of a rectangle P is denoted as *priority(P)* and corresponds to the x-coordinate of P^e.

Let us now discuss the handling of various types of events of a rectangle P of core segment s. At time t, let $r_1(t)$ and $r_2(t)$ be the rays of slope $+1$ and -1 respectively, emanating from $l(P^n) \cap l_t$, and $l(P^s) \cap l_t$ respectively, extending towards the left of l_t. Let $a(t)$ and $b(t)$ be the intersection points of $r_1(t)$ and $r_2(t)$ with the wavefront, respectively, at time t. In case of a wave collinear with $r_1(t)$ or $r_2(t)$, the rightmost endpoint is assigned to $a(t), b(t)$.

Consider time $t = \mathit{priority}(P)$. If the wavefront has not reached s yet, then the handling of the corresponding event (a start-vertical-rectangle or a horizontal-rectangle event) is similar to processing an ordinary line segment event [12, 9]: The portion of the wavefront between $a(t)$ and $b(t)$ is finalized and gets substituted by the wave-curve of P. There is one new action to take: For any crossing V-vertex on the finalized portion of the wavefront, induced by a rectangle Q, generate a V-event for the right V-vertex of $b_h(P, Q)$. If the wavefront has already covered portion of s, and P is vertical (start-vertical rectangle event) a V-event is generated to predict the first right V-vertex along s (if any). For this purpose, perform a binary search in the augmented wavefront to determine the wave between $a(t)$ and $b(t)$ with the rightmost w-max value as induced by a rectangle Q, and generate a V-event for the right V-vertex of $b_h(P, Q)$.

A V-event v is processed similarly. If at time $t = \mathit{priority}(v)$ the event is *invalid* i.e., the wavefront has already covered v, generate a new V-event. End-rectangle events are similar to right-events of [9].

A horizontal-rectangle event is processed at time $t = \mathit{priority}(P)$. The problem is to identify the intersections (V-vertices) of the wavefront with the vertical core segment s. To identify V-vertices efficiently we use the x-min value of the augmentation. Let r be the breakpoint of minimum *x-min* value between $a(t)$ and $b(t)$. If r is to the right of s then s must be entirely covered by the wavefront and there can be no intersections; $\mathit{reg}(P) = \emptyset$. Otherwise, trace the wavefront sequentially, starting at r, until the first intersections above and below r are determined. The intersection above (resp. below) corresponds to a *down* (resp. *up*) V-vertex v (resp. u). Repeat the process for the portions of the wavefront above v and below u until all intersections are determined.

The time complexity of the plane sweep algorithm, as presented, is $O(n + m + E) \log n)$, where E is the number of invalid V-events. There are two reasons for invalid V-events: 1. Potentially essential staircases of vertical rectangles whose crossings are not all essential. 2. Sequences of *strongly dominating* vertical rectangles, even in the case of non-crossing rectangles. Given a pair of vertical rectangles (P, Q), P is said to *dominate* Q if $Q^w < P^w$ and $Q^n < P^n$, $Q^s > P^s$. If in addition, there is a minimum enclosing square of Q that is crossing P, P is said to *strongly dominate* Q.

To eliminate source-2 of invalid V-vertex events, a preprocessing step may be added to the algorithm, which computes for every vertical rectangle P its *rightmost non-crossing rectangle* Q (if any) and a *potentially essential staircase* $\mathcal{R}$ to the right of Q^e (if

any). Q is the vertical rectangle, non-crossing with P, with the rightmost Q^w among all rectangles strongly dominated by P. This problem can be transformed into a *point dominance* problem in $\mathbb{R}^3$ and it can be solved by plane sweep in $O((n+V)\log n)$-time, as sketched in the following section. Thus, we conclude.

Theorem 3 *HVD(S) can be computed by plane sweep in $O((n+M)\log n)$ time, $O(n+M)$-space after a preprocessing step of $((n+V)\log n)$-time, $O(n+M)$-space. In the case of non-crossing rectangles this results in optimal $O(n\log n)$-time and $O(n)$-space.*

4 Preprocessing step – Point dominance

For any vertical rectangle P, map P^w into point $p = (P^n, -P^s, P^w)$ in $\mathbb{R}^3$. A point $p = (p_1, p_2, p_3)$ is said to *dominate* a point $q = (q_1, q_2, q_3)$ if $p_i \geqslant q_i$ for all $i, 1 \leqslant i \leqslant 3$.

In the non-crossing case, the problem is to determine, for every point p, the topmost point q dominated by p. For this purpose, we can sweep points in decreasing z-coordinate, while maintaining their *minima*, $M(t)$, i.e., the set of points above the sweeping plane at time t that do not dominate anything so far, projected on the xy-plane. $M(t)$ can be organized in a *priority search tree* $T_{M(t)}$ [8]. Let $p = (p_1, p_2, p_3)$ be the point to be considered at time $t = p_3$. Perform a range query on $T_{M(t)}$ for the NE quadrant of (p_1, p_2), i.e., range $J(p) = [p_1, \infty] \times [p_2, \infty]$. For any point r within $J(p)$, report (r, p). Delete all points in $M(t) \cap J(p)$ from $T_{M(t)}$ and insert (p_1, p_2). p can be deleted from $T_{M(t)}$ when the west edge of its leftmost minimum enclosing rectangle is reached. Since priority search trees are fully dynamic this is an $O(n\log n)$-time $O(n)$-space algorithm.

In the general case of crossing rectangles, in addition to $T_{M(t)}$, we also maintain a secondary priority search tree T'_t with all points that have already been matched with a point representing a crossing rectangle. Range queries in $T_{M(t)}$ remain the same, however, additional range queries are performed in T'_t as determined by points in $J(p)$. A point p gets deleted from T'_t when p is matched with a non-crossing rectangle or when its expiration time is reached. Details are given in the full paper.

References

[1] M. Abellanas, G. Hernandez, R. Klein, V. Neumann-Lara, and J. Urrutia. A combinatorial property of convex sets. *Discrete Computat. Geometry*, 17:307–318, 1997.

[2] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristán. The farthest color Voronoi diagram and related problems. In *Abstracts 17th European Workshop Comput. Geom. 2001*, 113–116.

[3] O. Cheong, H. Everett, M Glisse, J Gudmundsson, S. Hornus, S. Lazard, M. Lee, and H.-S. Na. Farthest-Polygon Voronoi Diagrams. arXiv:1001.3593v1 [cs.CG], 2010.

[4] F. Dehne and R. Klein. The big sweep": On the power of the wavefront approach to Voronoi diagrams. *Algorithmica*, 17:19–32, 1997.

[5] F. Dehne, A. Maheshwari, and R. Taylor. A coarse grained parallel algorithm for Hausdorff Voronoi diagrams. In *Proc. 2006 Int. Conf. on Parallel Processing*, 497–504, 2006.

[6] H. Edelsbrunner, L. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Discrete Computat. Geometry*, 4:311–336, 1989.

[7] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[8] E. M. McCreight. Priority Search Trees. *SIAM J. Comput.*, 14:257–276, 1985.

[9] E. Papadopoulou. Critical area computation for missing material defects in VLSI circuits. *IEEE Trans. Comp.-Aided Design*, 20(5):583–597, 2001.

[10] E. Papadopoulou. The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica*, 40:63–82, 2004.

[11] E. Papadopoulou. Net-aware critical area extraction for opens in VLSI circuits via higher-order Voronoi diagrams. *IEEE Trans. on Comp.-Aided Design*, to appear. Preliminary version in *LNCS* vol 4835, 716–727, 2007.

[12] E. Papadopoulou and D. T. Lee. The L_∞ Voronoi diagram of segments and VLSI applications. *Int. Journal of Computat. Geometry and Applications*, 11(5):503–528, 2001.

[13] E. Papadopoulou and D. T. Lee. The Hausdorff Voronoi diagram of polygonal objects: A divide and conquer approach. *Int. J. of Computational Geometry and Applications*, 14(6):421–452, 2004.

[14] D. P.Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete Computat. Geometry*, 267–291, 1993.

[15] J. Xu, L. Xu, and E. Papadopoulou. Computing the map of geometric minimal cuts. In *Proc. 20th Int. Symp. on Algorithms and Computation*, vol 5878 of *LNCS*, 244–254, 2009.

[16] "Voronoi CAA: Voronoi Critical Area Analysis", IBM CAD Tool, Dept. Electronic Design Automation, IBM Microelectronics, Burlington, VT. Initial patents: US6178539, US6317859. Distributed by Cadence.

Convex Treemaps with Bounded Aspect Ratio

Mark de Berg* Bettina Speckmann* Vincent van der Weele†

Abstract

Treemaps are a popular technique to visualize hierarchical data. The input is a weighted tree $\mathcal{T}$ where the weight of each node is the sum of the weights of its children. A treemap for $\mathcal{T}$ is a hierarchical partition of a rectangle into simply connected regions, usually rectangles. Each region represents a node of $\mathcal{T}$ and the area of each region is proportional to the weight of the corresponding node. An important quality criterium for treemaps is the aspect ratio of its regions. Unfortunately, one cannot bound the aspect ratio if the regions are restricted to be rectangles. Hence Onak and Sidiropoulos introduced *polygonal partitions*, which use convex polygons. We are the first to obtain convex partitions with optimal aspect ratio $O(\text{depth}(\mathcal{T}))$. We also consider the important special case that $\text{depth}(\mathcal{T}) = 1$, that is, single-level treemaps. We show how to construct convex single-level treemaps that use only four simple shapes for the regions and have aspect ratio at most 34/7.

1 Introduction

Treemaps are a popular technique to visualize hierarchical data [10]. The input is a weighted tree $\mathcal{T}$ where the weight of each node is the sum of the weights of its children. A treemap for $\mathcal{T}$ is a hierarchical partition of a rectangle into simply connected regions, usually rectangles. Each region represents a node of $\mathcal{T}$ and the area of each region is proportional to the weight of the corresponding node. To visualize the hierarchical structure the region associated with a node must contain the regions associated with its children. Shneiderman [11] first presented an algorithm for the automatic creation of rectangular treemaps. Treemaps are since used to visualize hierarchical data from a variety of application areas, for example, stock market portfolios [7], tennis competitions trees [6], large photo collections [3], and business data [13].

One of the most important quality criteria for treemaps is the aspect ratio of its regions [8]. Hence several approaches [3, 4] try to "squarify" the regions of a rectangular treemap. However, one cannot bound the aspect ratio if the regions are restricted to be rectangles. Hence, several types of treemaps using region shapes other than rectangles have been proposed. Balzer and Deussen [1, 2] use centroidal Voronoi tessellations. Their algorithm is iterative and can give no guarantees on the aspect ratio of the regions produced. Wattenberg [14] developed treemaps whose regions follow a space filling curve on a grid, so called Jigsaw maps. Jigsaw maps assume integer weights, which must add up to a square number. The regions of the maps are rectilinear, but highly non-(ortho)convex. However, they do have aspect ratio 4.

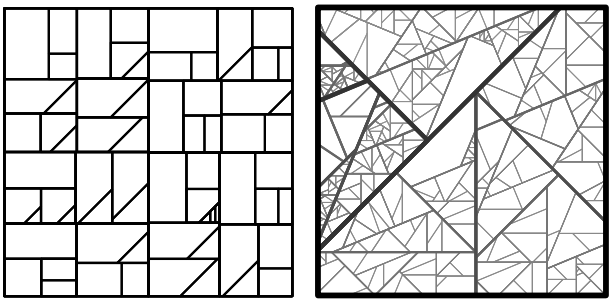


Figure 1: Treemaps constructed by our drawing algorithms: single-level convex, hierarchical convex.

Onak and Sidiropoulos [9] introduced *polygonal partitions*, which use convex polygons. They proved an aspect ratio of $O((\text{depth}(\mathcal{T}) \cdot \log n)^{17})$ for a tree $\mathcal{T}$ with n leaves. In cooperation with De Berg, this bound has since been improved to $O(\text{depth}(\mathcal{T}) + \log n)$ [5]. The latter paper also gives a lower bound construction of $\Omega(\text{depth}(\mathcal{T}))$.

Results and organization. We are the first to obtain convex partitions with optimal aspect ratio $O(\text{depth}(\mathcal{T}))$. Our recursive drawing procedure is described in Section 2. In Section 3 we also consider the important special case that $\text{depth}(\mathcal{T}) = 1$, that is, single-level treemaps. We show how to construct convex single-level treemaps that use only four shapes and have aspect ratio at most 34/7. Figure 1 shows two treemaps constructed by our drawing algorithms. In the treemap on the right the hierarchical structure is emphasized by line thickness and color: thicker, darker lines delimit nodes higher in the hierarchy. Specifically, there are four nodes on the top-level.

Preliminaries. Our input is a rooted tree $\mathcal{T}$. Following [5] we say that $\mathcal{T}$ is *properly weighted* if each node ν of $\mathcal{T}$ has a positive weight $\text{weight}(\nu)$ that equals the sum of the weights of the children of ν. We as-

*Department of Mathematics and Computer Science, TU Eindhoven, mdberg@win.tue.nl and speckman@win.tue.nl. B. Speckmann was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.022.707.

†Max-Planck-Institut für Informatik, Saarbrücken, vdweele@mpi-inf.mpg.de.

sume that weight(root($\mathcal{T}$)) = 1. A treemap for $\mathcal{T}$ associates a region $R(\nu)$ with each node $\nu \in \mathcal{T}$ such that (*i*) $R(\text{root}(\mathcal{T}))$ is the unit square, (*ii*) for every node we have area($R(\nu)$) = weight(ν), and (*iii*) for any node ν, the regions associated with the children of ν form a partition of $R(\nu)$. We denote the set of children of a node ν by children(ν).

The aspect ratio of a treemap is the maximum aspect ratio of any of its regions. For a single region, we use the following definition from [5]: the aspect ratio of a convex region R is $\text{diam}(R)^2/\text{area}(R)$.

Lemma 1 *Suppose all children of node ν have weight at most $2/3 \cdot \text{weight}(\nu)$. Then we can partition* children(ν) *into two subsets S_1 and S_2, such that* $\text{weight}(S_2) \leqslant \text{weight}(S_1) \leqslant 2/3 \cdot \text{weight}(\nu)$.

2 Hierarchical treemaps

We describe a recursive algorithm for computing a polygonal partition (convex treemap) of aspect ratio $O(\text{depth}(\mathcal{T}))$ for a properly weighted tree $\mathcal{T}$. Our algorithm has two phases. We first convert $\mathcal{T}$ into a binary tree $\mathcal{T}^*$ and then construct a partition for $\mathcal{T}^*$. This approach is similar to the one taken by De Berg *et al.* [5], but we implement both phases differently.

Converting to a binary tree. We recursively convert $\mathcal{T}$ into a binary tree $\mathcal{T}^*$, replacing each node with $k > 2$ children in $\mathcal{T}$ by a binary subtree with $k - 1$ nodes. During this process we assign a label $d(\nu)$ to each node ν, corresponding to the depth of ν in $\mathcal{T}$.

At every step, we treat a node ν with label $d(\nu)$ and convert the subtree rooted at ν. (Initially $\nu = \text{root}(\mathcal{T})$ with $d(\text{root}(\mathcal{T})) = 0$.) If ν is a leaf there is nothing to do. If ν has two children we recurse on these children and assign them label $d(\nu) + 1$. Otherwise ν has k children, children(ν) = $\{\nu_1, \ldots, \nu_k\}$, for some $k > 2$. We then distinguish two cases.

If there is a "heavy" child, say ν_1, with $\text{weight}(\nu_1) \geqslant \text{weight}(\nu)/2$, then we turn ν into a binary node whose children are ν_1 and a new node μ_1; the children of μ_1 are $\nu_2, \ldots, \nu_k$. We recurse on ν_1 and μ_1, with $d(\nu_1) = d(\nu) + 1$ and $d(\mu_1) = d(\nu)$. Otherwise all children have weight less than weight(ν)/2, and hence there is a partition of children(ν) into two subsets S_1 and S_2 such that $\text{weight}(S_i) \leqslant 2/3 \cdot \text{weight}(\nu)$ for $i \in \{1, 2\}$. We turn ν into a binary node with children μ_1 and μ_2, with children from S_1 and S_2, respectively, and we recurse on μ_1 and μ_2 with $d(\mu_1) = d(\mu_2) = d(\nu)$.

Drawing a binary tree. Generalizing ϕ-separated polygons [5], we define a (k, ϕ)-*polygon* to be a convex polygon P such that

(i) P does not have parallel edges, except possibly two horizontal edges and two vertical edges. Moreover, each non-axis-parallel edge e makes an angle of at least ϕ with any other edge and also with the x-axis and the y-axis.

(ii) If P has two horizontal edges, then $\text{diam}(P)/\text{height}(P) \leqslant k$.

(iii) If P has two vertical edges, then $\text{diam}(P)/\text{width}(P) \leqslant k$.

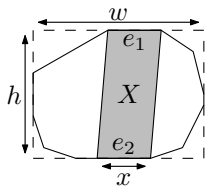
It follows from the definition of ϕ-separated polygons that a (k, ϕ)-polygon P is a ϕ-separated polygon, if it respects the following:

- if P has two horizontal edges, then $\text{height}(P) \geqslant \text{width}(P)$;
- if P has two vertical edges, then $\text{width}(P) \geqslant \text{height}(P)$.

Note that a (k, ϕ)-polygon P is ϕ-separated if its bounding box is square.

Lemma 2 *Any (k, ϕ)-polygon has aspect ratio* $O(\max(k, 1/\phi))$.

Proof. Let P be a (k, ϕ)-polygon with $w = \text{width}(P)$ and $h = \text{height}(P)$. Assume that $w \geqslant h$. Let e_1 and e_2 be the horizontal edges (possibly of length 0), let $x = \min(|e_1|, |e_2|)$, and let X be a parallellogram of width x. We distinguish two cases.



Case 1: $x > w/2$. P has two horizontal edges, so $h \geqslant \text{diam}(P)/k$. Clearly, the area of P is at least the area of X which is $xh > w \cdot \text{diam}(P)/(2k)$. The diameter of P is at most the diameter of the enclosing rectangle, hence $\text{diam}(P) \leqslant \sqrt{w^2 + h^2} \leqslant w\sqrt{2}$. Combined:

$$\text{asp}(P) = \frac{\text{diam}(P)^2}{\text{area}(P)} \leqslant \frac{2k \cdot \text{diam}(P)}{w} \leqslant 2\sqrt{2}k = O(k).$$

Case 2: $x \leqslant w/2$. We obtain polygon P' from P by reducing the length of e_1 and e_2 with $\min(x, w - h)$. Clearly, $\text{area}(P') \leqslant \text{area}(P)$. Observe that P' is a ϕ-separated polygon since either it has at most 1 horizontal edge (and $w - x \geqslant h$), or the bounding box of P' is square. Therefore, $\text{asp}(P') = O(1/\phi)$ [5]. Using $\text{diam}(P) \leqslant \sqrt{2}w$ and

$$\text{diam}(P') \geqslant w - \min(x, w - h) \geqslant w - x \geqslant w/2 \ ,$$

we calculate

$$\begin{aligned}\text{asp}(P) = \frac{\text{diam}(P)^2}{\text{area}(P)} &\leqslant \frac{2w^2}{\text{area}(P)} \\ &\leqslant 8 \cdot \frac{\text{diam}(P')^2}{\text{area}(P')} = 8 \cdot \text{asp}(P') = O(1/\phi).\end{aligned}$$

□

We construct the partition for $\mathcal{T}^*$ in a top-down manner. Each node ν in $\mathcal{T}^*$ has an associated region $R(\nu)$; initially $\nu = \text{root}(\mathcal{T}^*)$ and $R(\nu)$ is the unit square. We write $n(\nu)$ for the number of non-axis-parallel edges in $R(\nu)$. We maintain the following invariants:

(Inv-1) $n(\nu) \leqslant d(\nu) + 4$;

(Inv-2) $R(\nu)$ is a $(k, \phi(\nu))$-separated polygon for $k = \sqrt{17}$ and $\phi(\nu) = \pi/(2(d(\nu)+6))$.

The invariants are satisfied for $\nu = \text{root}(\mathcal{T}^*)$. Now consider a node ν that is not $\text{root}(\mathcal{T}^*)$. If ν is a leaf, there is nothing to do. Otherwise, let ν_1 and ν_2 be the two children of ν. Assume that $\text{weight}(\nu_1) \geqslant \text{weight}(\nu_2)$. We distinguish two cases.

Case 1: $d(\nu_1) = d(\nu) + 1$**.** Since $R(\nu)$ uses at most $d(\nu) + 4$ non-axis-parallel edges, there is a line ℓ that makes an angle of at least $\pi/(2(d(\nu) + 6))$ with each of the edges of $R(\nu)$ and with the x- and the y-axis. Imagine placing ℓ such that it splits $R(\nu)$ into two halves of equal area, and let R' be the half with the smallest number of non-axis-parallel edges. Now partition $R(\nu)$ into subpolygons $R(\nu_1)$ and $R(\nu_2)$ of the appropriate area with a cut c that is parallel to ℓ such that $R(\nu_2) \subset R'$. (Thus c lies inside R'.) We claim that both $R(\nu_1)$ and $R(\nu_2)$ satisfy the invariants.

Clearly $R(\nu_1)$ uses at most one edge more than $R(\nu)$. Since $d(\nu_1) = d(\nu)+1$, this implies that (Inv-1) is satisfied for $R(\nu_1)$. Now consider the number of non-axis-parallel edges of $R(\nu_2)$. This is no more than the number of non-axis-parallel edges of R'. At most two non-axis-parallel edges are on both sides of ℓ, hence this number is bounded by

$$n(\nu_2) \leqslant \left\lfloor \frac{n(\nu)+2}{2} \right\rfloor + 1 \leqslant \left\lfloor \frac{d(\nu)+6}{2} \right\rfloor + 1 = \left\lfloor \frac{d(\nu)}{2} \right\rfloor + 4 \leqslant d(\nu) + 4 \leqslant d(\nu_2) + 4.$$

Given the choice of ℓ, and because $d(\nu_i) \geqslant d(\nu)$ and $R(\nu)$ satisfies (Inv-2), we know that the minimum angle between any two non-parallel edges of $R(\nu_i)$ ($i \in \{1, 2\}$) is at least $\pi/(2(d(\nu_i) + 6))$. The following lemma, that we prove in the full version, suffices to show that $R(\nu_1)$ and $R(\nu_2)$ satisfy (Inv-2).

Lemma 3 *If $R(\nu_i)$ has two horizontal edges, then $\text{diam}(R(\nu_i))/\text{height}(R(\nu_i)) \leqslant k$ and if $R(\nu_i)$ has two vertical edges, then $\text{diam}(R(\nu_i))/\text{width}(R(\nu_i)) \leqslant k$, for $i \in \{1, 2\}$.*

Case 2: $d(\nu_1) = d(\nu)$**.** By construction of $\mathcal{T}^*$, $1/3 \cdot \text{weight}(\nu) \leqslant \text{weight}(\nu_1) \leqslant 2/3 \cdot \text{weight}(\nu)$. We now partition $R(\nu)$ into two subpolygons of the appropriate area with an axis-parallel cut orthogonal to the longest side of the axis-parallel bounding box of $R(\nu)$. The possible positions of this cut are limited by convexity, as specified in the following lemma.

Lemma 4 *Let P be a convex polygon with $\text{width}(P) \geqslant \text{height}(P)$. We can partition P with a vertical cut into two subpolygons P_1, P_2, where $\text{area}(P)/3 \leqslant \text{area}(P_i) \leqslant 2/3 \cdot \text{area}(P)$ (for $i \in \{1, 2\}$), such that $\text{width}(P)/4 \leqslant \text{width}(P_i) \leqslant 3/4 \cdot \text{width}(P)$.*

The number of non-axis-parallel edges of $R(\nu_1)$ and $R(\nu_2)$ is no more than the number of non-axis-parallel edges of $R(\nu)$. Since $d(\nu_i) \geqslant d(\nu)$, this implies $R(\nu_1)$ and $R(\nu_2)$ satisfy (Inv-1). As for (Inv-2), note that the cut does not introduce any new non-axis-parallel edges. It is thus met by the following lemma.

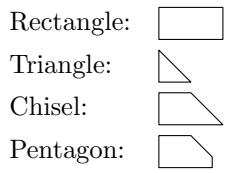
Lemma 5 *If $R(\nu_i)$ has two horizontal edges (for $i \in \{1, 2\}$), $\text{diam}(R(\nu_i))/\text{height}(R(\nu_i)) \leqslant \sqrt{17}$. Similarly, if $R(\nu_i)$ has two vertical edges, $\text{diam}(R(\nu_i))/\text{width}(R(\nu_i)) \leqslant \sqrt{17}$.*

Lemma 2, together with the fact that $\max_{\nu \in \mathcal{T}^*} d(\nu) = \text{depth}(\mathcal{T})$ and Inv-2, implies the result.

Theorem 6 *Every properly weighted tree of depth d can be represented by a polygonal partition (convex treemap) which has aspect ratio $O(d)$.*

3 Single-level treemaps

We now consider the special case that $\text{depth}(\mathcal{T}) = 1$. Our input is hence a set of positive weights. We describe a recursive drawing procedure that creates a treemap of aspect ratio at most 34/7 and uses only the four shapes depicted on the right. We do not recurse on pentagons, these are used only for single high weights.



We denote the bounding rectangle of a region R by $\rho(R)$. The aspect ratio of a rectangle ρ is defined as $\text{long}(\rho)/\text{short}(\rho)$, where $\text{long}(\rho)$ is the maximum of $\text{width}(\rho)$ and $\text{height}(\rho)$ and $\text{short}(\rho)$ is the minimum. This is equivalent to $\text{long}(\rho)^2/\text{area}(\rho)$. We write $\text{short}(R)$ for $\text{short}(\rho(R))$ and $\text{long}(R)$ for $\text{long}(\rho(R))$. We frequently use the aspect ratio of the bounding rectangle of a region and write $\text{asp}_\rho(R)$ for $\text{long}(R)/\text{short}(R)$. Our drawing procedure keeps the following invariant:

(Inv) $\text{asp}_\rho(R) \leqslant 4$ for all regions R.

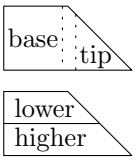
We convert $\mathcal{T}$ into a binary tree $\mathcal{T}^*$ as before and create a drawing for $\mathcal{T}^*$ in a top-down manner. Each node ν has again an associated region $R(\nu)$; initially $\nu = \text{root}(\mathcal{T}^*)$ and $R(\nu)$ is the unit square. If ν is a leaf, we are done. Otherwise, let ν_1 and ν_2 be the children of ν and let ν_1 be the heavier child. We distinguish three cases according to the shape of $R(\nu)$.

Case 1: Rectangle. If $\text{weight}(\nu_1)$ and $\text{weight}(\nu_2)$ are roughly equal, $\text{weight}(\nu)/(\text{asp}_\rho(R(\nu)) \cdot \text{weight}(\nu_2)) \leqslant 4$, we cut $R(\nu)$ through its longer side into two rectangles and have $\text{short}(R(\nu))^2/\text{area}(R(\nu_2)) \leqslant 4$ and hence $\text{asp}_\rho(R(\nu_i)) \leqslant 4$. Otherwise we draw $R(\nu_2)$ as a equilateral right-angled triangle in a corner of $R(\nu)$. Since we use only equilateral right-angled triangles the

bounding rectangle of any triangle is a square. Furthermore, since $\rho(R(\nu_1))$ equals $\rho(R(\nu)$ we also have $\text{asp}_\rho(R(\nu_1)) \leqslant 4$.

Case 2: Triangle. We cut a triangle into a chisel and a triangle. The bounding rectangle of the chisel has aspect ratio at most 4 if its width is at least 1/4 times the width of the triangle. This is the case if $\text{weight}(\nu_1)/\text{weight}(\nu) \geqslant 7/16$. Since ν_1 is the heavier child, this always holds.

Case 3: Chisel. Cutting the longer side of a chisel, by an *orthogonal cut*, yields two regions *base* and *tip*. Cutting the shorter side of a chisel, by a *parallel cut*, yields regions *higher* and *lower*. We first give two lemmas for analyzing the two types of cuts. For brevity, we write $\text{rel}(\nu_i) = \text{weight}(\nu_i)/\text{weight}(\nu)$, $i \in \{1, 2\}$.



Lemma 7 *For an orthogonal cut, the regions meet the invariant if* $\text{short}(R(\nu))^2/\,\text{area}(\textit{base}) \leqslant 4$.

Lemma 8 *For a parallel cut, with higher* $= R(\nu_1)$, *we have* $\text{asp}_\rho(\textit{lower}) \leqslant \text{asp}_\rho(\textit{higher})$ *iff* $\text{rel}(\nu_1) \leqslant \text{asp}_\rho(R(\nu))/(2 \cdot \text{asp}_\rho(R(\nu)) - 1)$.

We distinguish three cases.

Case 1: ν_1 is a leaf. We let $R(\nu_1)$ be the base and $R(\nu_2)$ be the tip of an orthogonal cut. Since $\text{rel}(\nu_1) \geqslant 1/2$, we can easily show that $\text{short}(R(\nu))^2/\,\text{area}(\text{base}) \leqslant 4$. By Lemma 7, the subregions meet the invariant.

Case 2: ν_1 is not a leaf and $\text{asp}_\rho(R(\nu)) \geqslant 3/2$. We let $R(\nu_1)$ be the tip and $R(\nu_2)$ be the base of an orthogonal cut. The base is certainly a rectangle, since the rectangular part of the chisel is at least half of its total area and ν_2 is the lighter child. Using $\text{rel}(\nu_2) \geqslant 1/3$ (Lemma 1), we can show that $\text{short}(R(\nu))^2/\,\text{area}(\text{base}) \leqslant 4$. Then, the subregions meet the invariant by Lemma 7.

Case 3: ν_1 is not a leaf and $\text{asp}_\rho(R(\nu)) < 3/2$. We let $R(\nu_1)$ be the higher and $R(\nu_2)$ be the lower of a parallel cut. It is not hard to see that $\text{asp}_\rho(\text{higher})$ meets the invariant if $\text{short}(\text{higher}) \geqslant 3/8 \cdot \text{short}(R(\nu))$, which follows from ν_1 being the heavier child. Moreover, since $\text{rel}(\nu_1) \leqslant 2/3$ by Lemma 1, it follows from Lemma 8 that lower meets the invariant as well.

The aspect ratio of a region R, compared to $\text{asp}_\rho(R)$, is maximal if R is a chisel. Together with the invariant, this implies the following result.

Theorem 9 *Every properly weighted single-level tree can be represented by a convex treemap which uses only four simple shapes and has aspect ratio at most* 34/7.

References

[1] M. Balzer and O. Deussen. Voronoi treemaps. In *Proc. IEEE Symposium on Information Visualization*, pages 7–14, 2005.

[2] M. Balzer, O. Deussen, and C. Lewerentz. Voronoi treemaps for the visualization of software metrics. In *Proc. ACM Symposium on Software Visualization*, pages 165–172, 2005.

[3] B. B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Transactions on Graphics*, 21(4):833–854, 2002.

[4] M. Bruls, K. Huizing, and J. van Wijk. Squarified treemaps. In *Proc. Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42. Springer, 2000.

[5] M. de Berg, K. Onak, and A. Sidiropoulos. Fat polygonal partitions with applications to visualization and embeddings. In preparation. `http://arxiv.org/abs/1009.1866v1`, 2010.

[6] L. Jin and D. C. Banks. Tennisviewer: A browser for competition trees. *IEEE Computer Graphics and Applications*, 17(4):63–65, 1997.

[7] W. Jungmeister and D. Turo. Adapting treemaps to stock portfolio visualization. Technical report UMCP-CSD CS-TR-2996, University of Maryland, 1992.

[8] N. Kong, J. Heer, and M. Agrawala. Perceptual guidelines for creating rectangular treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):990–998, 2010.

[9] K. Onak and A. Sidiropoulos. Circular partitions with applications to visualization and embeddings. In *Proc. 24th Symposium on Computational Geometry*, pages 28–37, 2008.

[10] B. Shneiderman. Treemaps for space-constrained visualization of hierarchies. `http://www.cs.umd.edu/hcil/treemap-history/index.shtml`.

[11] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[12] E. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 2001.

[13] R. Vliegen, J. J. van Wijk, and E.-J. van der Linden. Visualizing business data with generalized treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):789–796, 2006.

[14] M. Wattenberg. A note on space-filling visualizations and space-filling curves. In *Proc. IEEE Symposium on Information Visualization*, pages 181–185, 2005.

A parallel GPU-based algorithm for Delaunay edge-flips

Cristobal A. Navarro* Nancy Hitschfeld-Kahler† Eliana Scheihing*

Abstract

The edge-flip technique can be used for transforming any existing triangular mesh into one that satisfies the Delaunay condition. Although several implementations for generating Delaunay triangulations are known, to the best of our knowledge no full parallel GPU-based implementation just dedicated to transform any existent triangulation into a Delaunay triangulation has been reported yet. In the present work, we propose a two-phase iterative GPU-based algorithm, that transforms any 2D planar triangulations and 3D triangular surface meshes into their respective Delaunay form. We tested our method with meshes of different size, and compared it with a sequential CPU-implementation. Based on these results, our algorithm strongly improves the performance with respect to a classic CPU-implementation, thus making it a good candidate for interactive/real-time applications.

1 Introduction

Delaunay triangulations are widely used in several applications because they present good properties that make them useful in 2D and 3D numerical simulations. In the last two decades there has been a considerable amount of work done on computing Delaunay triangulations, from different sequential implementations [9] to recently parallel ones [1], and in particular, GPU-based methods [8, 2]. These works belong to the case when a Delaunay triangulation needs to be computed from a given set of points or from an initial geometry and not from an existing triangulation. On the other hand, transformation methods for existing triangulations have relied on the edge-flip technique first introduced by Lawson [7]. Though the edge-flip technique is simple and the implementation is straightforward, the order of the algorithm is $O(n^2)$ in the worst case [6, 5] (where n refer the number of points of the triangulation) making its performance potentially slow for millions of triangles. The algorithm of Rong et al. [8] uses the edge-flip operation, but it is implemented on the CPU. Very recently, they improved their method by implementing all the operations in parallel [2]. The main differences with our approach are that our algorithm is more straightforward and simple since it is only dedicated to transform any triangulation into a Delaunay triangulation. In addition, Cao's algorithm uses one thread per triangle and needs to update more complex data structures. Our algorithm uses on thread per-edge and the used data structures are oriented for real-time rendering. There is also another interesting article that describes a parallel GPU-based algorithm designed to generate triangulations for image reconstruction [3]. This algorithm performs edge-flips in parallel according to a function cost that depends on the treated image. If the function is the Delaunay condition, then their algorithm could be used for the purpose of our paper. Their approach is also iterative and uses one thread per edge, but uses different data structures which are oriented to store regions of influence of each edge e and the implementation is done with shaders.

The application that motivates our work is the simulation of stem deformations. At each time, the geometry of the triangulation changes and we want to restore the Delaunay condition. Then we want a restoration method capable to perform close to interactive/real-time levels, executing on a desktop workstation. The contribution of this work is a GPU-based method that improves 2D and 3D triangulations. This short paper is organized as follows: Section 2 describes the data structures and Section 3 covers the algorithm and some implementation details. In Section 4 we present results from different tests, to finally discuss and conclude our work in Section 5 .

2 Involved data structures

Proper data structures have been defined to represent a triangulation in order to use as efficiently as possible the GPU's architecture. This representation is inspired on the Dynamic Render Mesh [10]. Figure 1 illustrates the three main components: Vertices, Triangles and Edges.

Vertices are handled via the Vertex array where each element contains a position (x, y) or (x, y, z) depending on the dimension used. The Triangles array is a set of indices to the vertex array where each three consecutive indices corresponds to a triangle. Each edge contains a pair of vertex indices v_1, v_2 and two references t_a, t_b to the triangles that share it (for boundary edges, t_b remains unused). Both t_a and t_b contain

*Informatics Institute, Austral University, Chile, axischire@gmail.com, escheihi@inf.uach.cl

†Department of Computer Science, FCFM, University of Chile, Chile, nancy@dcc.uchile.cl

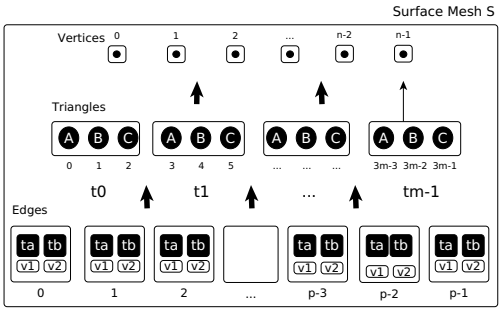


Figure 1: Data structures for mesh rendering/processing.

a pair of indices that point to the exact indices on the Triangles array where this edge can be found. In other words, every edge can access its vertex data directly through v_1, v_2 or indirectly via t_a, t_b. This redundant information becomes useful for consistency checking after an edge flip is performed nearby. The data model was designed so that it could be naturally implemented and integrated with the OpenGL API and at the same time implementable on the CUDA [4] architecture.

3 Algorithm Overview

Each iteration of the proposed algorithm is divided in two phases: (1) Exclusion & Processing and (2) Repair. The algorithm finishes when all edges fulfill the Delaunay condition.

3.1 Exclusion & Processing

This phase is in charge of a double work: selecting and flipping edges in parallel. It begins by testing each edge against the Delaunay condition. If the condition is fulfilled, then e is Delaunay and the execution thread ends ($d = 1$), otherwise the thread continues alive ($d = 0$). Figure 2 shows an example of a small mesh (S_e), where edge e does not satisfy Delaunay condition. In this figure, edge e is the unique internal edge of the mesh, thus the only one to be analyzed (boundary edges are never flipped). This part of the phase adjusts perfectly to the GPU architecture because each edge is an independent problem that can be assigned to a unique thread.
Because of neighbor dependency problems, it is not always possible to flip the complete set of $d = 0$ edges in one iteration because the flip of a given edge e produces a transformation on the triangles (t_i, t_j) where this edge belongs to. This transformation affects directly the neighborhood information of the other edges of the triangles that share e, making impossible to flip those while e is being flipped. However, it is

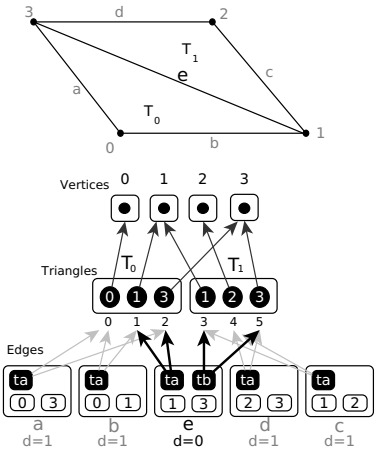


Figure 2: On this mesh, e is the only one to flip.

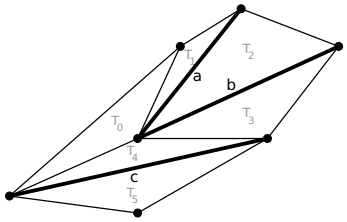


Figure 3: The edge subset can either a,c or b,c.

possible to process a subset of edges A that fulfill the following condition:

$$\forall\, e_1, e_2 \in A \quad T_{e_1} \cap T_{e_2} = \emptyset \text{ where } T_e = \{t \in T : e \in t\} \tag{1}$$

In order to get a proper subset, we propose a parallel exclusion mechanism in which each thread managing a non-Delaunay edge (non-Delaunay edge thread), will be allowed to flip its edge if it is the first one to make a state transition from "free" to "taken" (binary flag) on the two triangles that share it. Non-Delaunay edge threads that could not catch their two triangles will find the opportunity to flip their edge on future iterations. Atomic operations are used in this exclusion mechanism in order to avoid any race condition. Figure 3 shows a mesh, where edges a, b, and c need to be flipped but they cannot be processed at the same time. After applying condition 1, the resulting subset can be either $\{a, c\}$ or $\{b, c\}$ but a and b will never be selected together, because they share T_2. We design the per thread edge-flip method as an index exchange between the two triangles related to the processed edge e. This transformation can be seen as a rotation of the involved triangles fully independent from the rest of the mesh. The transformation is done on the Triangles array using the information t_a and t_b stored in the Edges array by following the next steps:

- Find the opposite vertex indices u_1 and u_2 of e in the Triangles array going through t_a and t_b (Edges array).
- Locate the position of the first common vertex index c_1 in the Triangles array going through t_a.
- Locate the position of the second common vertex index c_2 in the Triangles array going through t_b.
- Exchange data, by copying u_1 into Triangles[c_2], and u_2 into Triangles[c_1].
- Update the values of t_a, t_b and v_1, v_2 related to e in the Edges array.

Figure 4 illustrates the case of flipping the edge e from the simple mesh S_e (Figure 2) and how the rotation transformation is achieved.

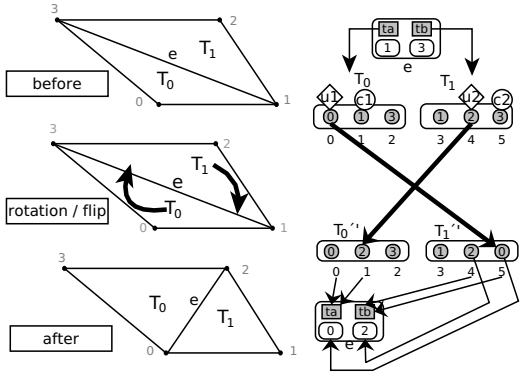


Figure 4: The edge-flip as a rotation of triangles.

3.2 Repair

After the parallel edge-flips, consistency problems might appear on the edges nearby a flipped edge. More specifically, some neighbor edges can now store references to triangles whom they do not belong anymore (obsolete t_a, t_b indices). We say that an edge is inconsistent when its vertex indices obtained through t_a, t_b differ from the ones stored in v_1 and v_2 which will always be the correct values. Therefore the edges remain in the same original place but might belong to different triangles t_a, t_b. In the following simple mesh S_e, inconsistent information appears at edges b and d right after flipping e (Figure 5). The detection and fixing of inconsistent edges can also be achieved in parallel; Given an edge e, just compare the vertex indices accessed via t_a and t_b against v_1 and v_2. If the values are different, then search the correct values in the indices of the triangle that were rotated together with t_a and t_b. The information of the triangles that were rotated together can be stored in the "Exclusion & Processing" phase as an array R[] where R[t_i] = t_j and vice-versa.

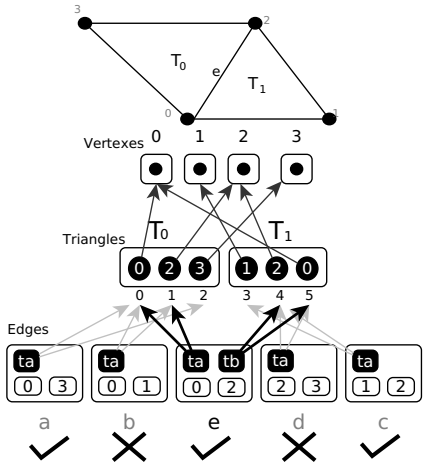


Figure 5: Edges marked with a cross are inconsistent.

3.3 Handling problematic cases

During the first phase, there are two cases that we should explain in more detail: (1) the handling of co-circular configurations and (2) if it is possible that a dead-lock might occur for some special cases.

We solve the case (1) using an ϵ value. For each edge e, our algorithm uses the simple Delaunay routine that computes the angles λ and γ opposite to e from the triangles that share e and checks the following condition:

$$\lambda + \gamma \leq \pi + \epsilon \qquad (2)$$

If the condition is true, the edge e fulfills the Delaunay condition. Otherwise is a candidate to be flipped. Since our algorithm does not flip edges whose triangles are defined by co-circular or almost co-circular vertices, each chain of edge-adjacent triangles forming a cycle must have an edge that fulfill the Delaunay condition. This is the smallest edge of the chain. Then, in a chain like this there will be at least one edge than can be flipped: one of the edges that belongs to one of the two triangles that share the smallest edge. We are working on a formal demonstration for this case.

4 Evaluation and results

The hardware used for testing our algorithm consists of an AMD Phenom Quad 2.7GHZ CPU, and two GPUs: Geforce 9800GTX+ and GTX 580. In the following, we call our method MDT (Massive Delaunay Transformations). Tests were done with both 2D and 3D surface triangulations. Figure 6 shows the performance of the algorithm for 2D random generated triangulations of a quad but with different number of triangles. These random triangulations were generated using Blender (subdivision and noise). We also added interactive and real-time time thresholds as good performance reference values. On the

9800GTX+, the performance is good for meshes of less than one million triangles but not so good for larger meshes. On the other hand, the GTX 580 performs considerably better as expected, making possible real-time Delaunay transformations on very large triangulations. We have also evaluated MDT with 3D

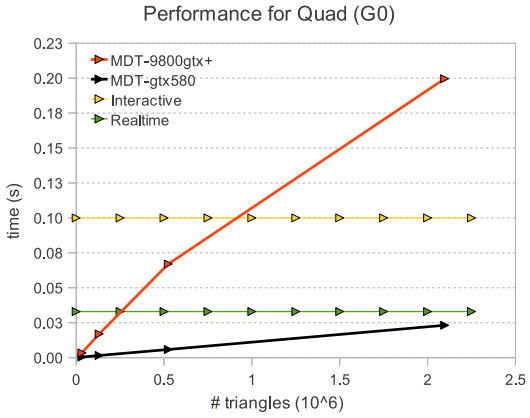


Figure 6: Performance on different architectures.

surface triangulations such as the dragon (Stanford Computer Graphics Laboratory), the horse (Cyberware Inc), a moai (Geomview) and the infinite built with our custom tools. Table 1 compares the MDT running on the GTX 580 with a locally built sequential CPU-implementation.

Table 1: Performance comparison of 3D surface meshes.

Mesh	# edges	CPU-Method [s]	MDT [s]
Moai	30,000	0.867	0.001
Horse	337,920	15.942	0.012
Dragon	1,309,256	0.387	0.046
Infinite	4,758,912	274.1	0.157

5 Discussion and Conclusions

Our preliminary results show that MDT can become a practical and useful algorithm for modern dynamic and interactive/real-time applications that need to handle all time a Delaunay mesh. Our GPU-based solution scales very well on newer GPU architectures. The algorithm has also no deep complexity and the data structures are 100% compatible with modern Graphics APIs. MDT is best suited for large meshes composed by at least thousands of edges and most of the times the transformation is achieved with few iterations. We have also tested MDT with one of the worst case meshes for the sequential algorithm [5]. In this case, MDT presents a particular behavior maximizing the number of parallel edge-flips in the middle iterations instead of in the first ones as with the other tests. In the near future, we want to analyze precisely the behavior of our algorithm and to compare its performance with the approaches mentioned above.

6 Acknowledgments

We would like to thank to the anonymous reviewers of the ACM ToG journal and EuroCG2011 for the very useful comments on how to improve our work.

References

[1] C. Antonopoulos, X. Ding, A. Chernikov, F. Flagojevic, D. Nikolopoulos, and N. Chrisocoides. Multigrain parallel delaunay mesh generation: challenges and opportunities for multithreaded architectures. In *ICS '05 proceedings*, pages 367–376, New York, NY, USA, 2005. ACM.

[2] T. T. Cao. Computing 2d delaunay triangulation using gpu. *Manuscript in preparation*, 2010. http://www.comp.nus.edu.sg/ tants/delaunay2D Download.html.

[3] M. Cervenanský, Z. Tóth, J. Starinský, A. Ferko, and M. Srámek. Parallel gpu-based data-dependent triangulations. *Computers & Graphics*, 34(2):125–135, 2010.

[4] N. Corporation. *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide*, 2007.

[5] H. Edelsbrunner. Geometry and topology for mesh generation (cambridge monographs on applied and computational mathematics), 2001.

[6] S. Fortune. A note on delaunay diagonal flips. *Pattern Recognition Letters*, 14(9):723 – 726, 1993.

[7] C. L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365 – 372, 1972.

[8] G. Rong, T.-S. Tan, T.-T. Cao, and Stephanus. Computing two-dimensional delaunay triangulation using graphics hardware. In *I3D '08 proceedings*, pages 89–97, New York, USA, 2008. ACM.

[9] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *First Workshop on Applied Computational Geometry*, pages 124–133. ACM, 1996.

[10] R. F. Tobler and S. Maierhofer. *A Mesh Data Structure for Rendering and Subdivision*. Plzen, Czech Republic, 2006.

A Competitive Strategy for Distance-Aware Online Shape Allocation

Sándor P. Fekete* Nils Schweer* Jan-Marc Reinhardt*

Abstract

We consider the following online problem, motivated by grid computing: Given an $N \times N$ square S, and a sequence of numbers n_i with $\sum_{j=0}^{i} n_j \leq N^2$; at each step i, select a region C_i of previously unassigned area n_i in S. The objective is to minimize the maximum average Manhattan distance between points from the same set C_i. We present a competitive online strategy, based on a thorough analysis of space-filling curves; for continuous shapes, we prove a factor of 1.8092, and 1.7849 for discrete shapes.

1 Introduction

Many optimization problems deal with allocating point sets to a given environment. Frequently, the problem is to find compact allocations, such that points from the same set are closely together. One natural measure is to consider the average distance between points. A practical example occurs in the context of grid computing, where one needs to assign a sequence of jobs i, each requiring n_i processors, to a subset C_i of nodes of a square grid, such that the average communication delay between nodes of the same job is minimized; as this delay corresponds to the number of grid hops [6], this motivates the geometric task of finding subsets with a small average Manhattan distance.

Even in an offline setting without any occupied nodes, finding an optimal allocation for a single set of size n_i is not an easy task; the results are typically "round" shapes. If a whole sequence of sets have to be allocated, packing such shapes onto the grid will produce gaps, causing later sets to become scattered, and thus lead to extremely bad average distances. Even packing rectangular shapes is not a remedy, as the resulting packing problem is NP-hard, and scattered allocations may still occur.

In this paper, we give a first algorithmic analysis for the *online* problem. Using an allocation scheme based on a space-filling curve, we establish competitive factors of 1.8092 and 1.7849 for minimizing the maximum average Manhattan distance within an allocated set.

*Algorithms Group, Braunschweig Institute of Technology, Germany, {s.fekete,n.schweer,j-m.reinhardt}@tu-bs.de

Related Work The problem of finding the "optimal shape of a city", i.e., a shape of unit area that minimizes the average distance, was first considered by Karp, McKellar, and Wong [4]; independently, Bender, Bender, Demaine, and Fekete [1] showed that this shape can be characterized by a differential equation for which no closed form is known. For the case of a finite set of n points that needs to be allocated to a grid, Demaine et al. [3] showed that there is an $O(n^{7.5})$ dynamic-programming algorithm. In an offline setting in the presence of occupied points, Bender et al. [2] gave some simple 1.75-approximation algorithms, and a polynomial-time approximation scheme. Space-filling curves for processor allocation have been used before [6]; in particular, Wattenberg [10] has proposed an allocation scheme similar to ours, for purposes of minimizing the maximum *diameter* of an allocated shape. However, neither paper has yielded algorithmic results for our problem, and no constant competitive factor was proven.

2 Preliminaries

We examine the problem of selecting shapes from a square, such that the maximum average L_1-distance of the shapes is minimized. We first formulate the problem more precisely.

Definition 1 *A* city *is a shape in the plane with fixed size. For a city C of size n we call*

$$c(C) = \frac{1}{2} \iiiint_{(x,y),(u,v)\in C} |x-u|+|y-v| \,\mathrm{d}v \,\mathrm{d}u \,\mathrm{d}y \,\mathrm{d}x$$

the Manhattan distance between all pairs of points in C *and*

$$\phi(C) = \frac{2\,c(C)}{n^{5/2}}$$

the ϕ-value *or* average distance *of* C.

We divide by $n^{2.5}$ to get a dimensionless measure for the average distance, see [1].

The problem For a sequence $n_1, n_2, \ldots, n_k \in \mathbb{R}^+$ with $\sum_{i=1}^{k} n_i \leq 1$, cities $C_1, C_2, \ldots, C_k$ are to be chosen from the unit square, such that $\max_{1\leq i\leq k} \phi(C_i)$ is minimized.

Although it has not been proven, the problem is assumed to be NP-hard, see [8]; if we restrict city shapes to be rectangles, there is an immediate reduction from

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

deciding whether a set of squares can be packed into a larger square [5]. (A special case arises from considering integers, which corresponds to choosing grid locations.) On the one hand, to make things a little easier, we only consider sizes n_i which are larger than a known lower bound ε. On the other hand, our approximation works online, i.e., we choose the cities in a specified order, and no changes are made to previously allocated cities.

There are lower bounds for $max_{1\leq i\leq k}\phi(C_i)$ that generally cannot be achieved by any algorithm. One important result is the following theorem.

Theorem 1 *Let C be any city. Then $\phi(C) \geq 0.650245$.*

A proof can be found in [1]. For $n_1 = 1$ any algorithm must select the whole unit square, thus 2/3, the ϕ-value of a square, is a lower bound. We will discuss approaches for a better bound in the end.

3 An Algorithm

While long and narrow shapes tend to have large ϕ-values, shapes that fill large parts of an enclosing rectangle with similar width and height usually have better average distances. Our approach uses the space-filling Hilbert curve to yield a provably constant competitive factor.

First, we divide the unit square into a grid consisting of $2^r \times 2^r$ cells, such that the size of each cell of the grid is equal to or less than the lower bound of the n_i. The Hilbert curve is based on a recursive construction scheme and becomes space-filling for infinite repetition of said scheme [7]. For a finite number r of repetitions, the curve traverses the used grid. For $1 \leq r \leq 3$, the curve is shown in figure 1. Thus, the Hilbert curve provides an order for the cells of the grid.

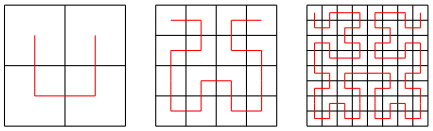


Figure 1: Hilbert curve with $1 \leq r \leq 3$

Furthermore, we denote the size of each cell by $c = 4^{-r}$, and the cell in row j and column k after r iterations by $E_r^2[j][k] = [2^{-r}(j-1), 2^{-r}j] \times [2^{-r}(k-1), 2^{-r}k]$. For the sake of concise presentation of our analysis within this short abstract, we assume that every input n_i is an integral multiple of c; in general, we can proceed to make the grid self-refining. The description of the algorithm is simple: for every input n_i we choose the next n_i/c cells traversed by the Hilbert curve as the city C_i, starting in the upper left corner of the grid.

A formal implementation of the curve can be done using text-rewriting rules, such as the ones in [9].

4 Analysis

Our analysis of the algorithm focuses on two points: describing a way to systematically find worst cases of a specified size and finding a competitive factor.

Definition 2 *We denote by W_i a city with i cells (or i whole $E_l^2[j][k]$ for $l \neq r$) that is a worst-case output of our strategy for cities of size i.*

As all cities generated by our strategy consist of an integral number of cells, the ϕ-value of a produced city can be calculated considering only the distances and count of the cells occupied in the grid. Thus, the W_i can be found by simply comparing the distances of all i-tuples of cells on the grid, which are successively traversed by the Hilbert curve. This leaves the question of how large the grid has to be chosen to contain a city W_i traversed by the curve.

Lemma 2 *For $r = \lceil \log_2(\lceil i/4 \rceil + 1) \rceil + 2$ the Hilbert curve traverses a city W_i.*

We do not give a proof here. It can be done using the text-rewriting rules from [9] to construct a formal grammar and show via induction that after a certain number of substitutions, every possible sequence of symbols of a particular length has been generated, i.e., the sequence corresponding to the city with the greatest ϕ-value must have been produced as well. We used this approach to determine the shapes and ϕ-values of the W_i for $i \leq 65$. The average distances for $16 \leq i \leq 63$ can be seen in the fourth column of table 1; the worst cases among the examined ones are W_{56} and W_{14}, which have the same shape, shown in figure 2.

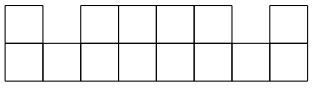


Figure 2: W_{14}

The worst cases can also be used to find an upper bound for the maximum average distance and ultimately give a competitive factor.

Lemma 3 *For every $0 \leq l \leq r$ the $E_l^2[j][k]$ are traversed by the Hilbert Curve in a specific order, i.e., the curve does not only pose an order on the cells of the grid, but on each set of the $E_l^2[j][k]$.*

Again, we do not give a formal proof here; the claim follows directly from the definition of the Hilbert curve, as its construction scheme is invoked recursively for sub-squares of the unit square.

i	$c^*(W_i)$	$c_{city}(i)$	$\phi(W_i)$	$\Phi(W_{i+2})$
16	$434\,^2/_3$	$338\,^2/_3$	0.8490	1.1764
17	$512\,^2/_3$	$396\,^1/_3$	0.8605	1.1497
18	$602\,^1/_3$	$457\,^1/_3$	0.8764	1.1174
19	685	$522\,^1/_3$	0.8706	1.1058
20	768	$591\,^2/_3$	0.8587	1.1098
21	870	663	0.8610	1.0903
22	$992\,^2/_3$	$749\,^1/_3$	0.8745	1.0713
23	$1101\,^2/_3$	$839\,^1/_3$	0.8685	1.0424
24	1216	933	0.8619	1.0150
25	$1322\,^1/_3$	$1032\,^1/_3$	0.8463	0.9777
26	1432	1134	0.8309	0.9701
27	$1527\,^2/_3$	1249	0.8066	0.9785
28	1672	$1365\,^1/_3$	0.8061	0.9864
29	$1853\,^1/_3$	1492	0.8184	0.9773
30	2046	$1622\,^2/_3$	0.8301	0.9710
31	2213	$1759\,^2/_3$	0.8272	0.9726
32	$2393\,^1/_3$	$1898\,^2/_3$	0.8263	0.9791
33	2602	2057	0.8319	0.9735
34	$2835\,^2/_3$	$2216\,^2/_3$	0.8414	0.9691
35	3045	2384	0.8403	0.9712
36	3266	$2554\,^2/_3$	0.8400	0.9772
37	$3519\,^1/_3$	$2727\,^2/_3$	0.8453	0.9726
38	$3799\,^1/_3$	$2921\,^2/_3$	0.8536	0.9682
39	$4049\,^2/_3$	$3117\,^2/_3$	0.8527	0.9570
40	$4309\,^1/_3$	3322	0.8517	0.9463
41	4545	$3530\,^1/_3$	0.8445	0.9307
42	4788	3749	0.8376	0.9214
43	5009	3976	0.8262	0.9306
44	$5266\,^2/_3$	$4205\,^1/_3$	0.8202	0.9393
45	$5641\,^1/_3$	$4456\,^2/_3$	0.8306	0.9393
46	$6031\,^1/_3$	4712	0.8405	0.9395
47	$6379\,^2/_3$	$4970\,^1/_3$	0.8425	0.9439
48	$6741\,^1/_3$	5234	0.8446	0.9505
49	$7147\,^1/_3$	$5507\,^1/_3$	0.8505	0.9512
50	$7586\,^1/_3$	5788	0.8583	0.9516
51	7993	$6076\,^1/_3$	0.8606	0.9559
52	$8411\,^1/_3$	6368	0.8628	0.9620
53	8878	$6691\,^2/_3$	0.8683	0.9619
54	$9379\,^1/_3$	$7017\,^1/_3$	0.8754	0.9617
55	9835	$7352\,^1/_3$	0.8768	0.9569
56	10304	7690	0.8781	0.9522
57	10733	$8033\,^2/_3$	0.8751	0.9445
58	$11173\,^1/_3$	$8384\,^2/_3$	0.8723	0.9372
59	$11583\,^2/_3$	$8749\,^1/_3$	0.8665	0.9268
60	$12005\,^1/_3$	$9117\,^1/_3$	0.8610	0.9225
61	12391	$9506\,^1/_3$	0.8527	0.9232
62	12862	$9904\,^2/_3$	0.8499	0.9201
63	13415	$10305\,^1/_3$	0.8517	0.9175

Table 1: Total and average distances for cities W_n allocated according to our strategy, as well as the optimal values $c_{city}(n)$ according to [3].

Lemma 4 *Let C be a city generated by our strategy with size $n \leq k\,4^j\,c$ for $j \in \{0, 1, \ldots, r\}, k \in \mathbb{N}$. Furthermore, let W be a city W_{k+1} of size $(k+1)\,4^j\,c$. Then $c(C) \leq c(W)$ holds.*

Proof. C cannot contain parts of more than $k+1$ of the sub-squares $E^2_{r-j}[p][q]$. Otherwise we would have a contradiction to Lemma 3, as it would contain fewer than $k-1$ complete sub-squares, thus implying that they are not traversed in order.

Consequently, C can be bounded by a city X consisting of $k+1$ sub-squares of size $4^j\,c$, and $c(C) \leq c(X)$ holds. As there is no city consisting of $k+1$ sub-squares with a greater ϕ-value than W_{k+1}, $c(X) \leq c(W)$ holds as well. Combining both inequalities yields $c(C) \leq c(W)$. □

Theorem 5 *Our strategy guarantees* $\max_{1 \leq i \leq k} \phi(C_i) \leq 1.1764$.

Proof. For $r \leq 2$ the only possible inputs n are $c \leq n \leq 16c$. In this case, $\phi(C)$ can be bounded by $\max_{1 \leq i \leq 16} \phi(W_i) = \phi(W_{14}) \approx 0.8781$.

In the following consider $r > 2$ and $4^j c < n \leq 4^{j+1}c$ for $j \in \{2, \ldots, r-1\}$.

For each of the cases $l\,4^{j-2}c < n \leq (l+1)\,4^{j-2}c$ with $l \in \{16, 17, \ldots, 63\}$ we can use Lemma 4 to get a city W of size $(l+2)\,4^{j-2}\,c$ and shape W_{l+2}, such that $c(C)$ is less than or equal to $c(W)$.

Thus, we get the following inequality:

$$\phi(C) \leq \frac{2\,c(W)}{(l\,4^{j-2}c)^{5/2}}$$

Because of the definition of the ϕ-value, this yields

$$\frac{\phi(W_{l+2})((l+2)\,4^{j-2}c)^{5/2}}{(l\,4^{j-2}c)^{5/2}} = \phi(W_{l+2})\left(1+\frac{2}{l}\right)^{5/2}$$

We denote $\phi(W_{l+2})\left(1+\frac{2}{l}\right)^{5/2}$ by $\Phi(W_l)$ and list the values of $\Phi(W_l)$ for $16 \leq l \leq 63$ in the fifth column of table 1. Therefore, $\phi(C) \leq 1.1764$ holds. □

Corollary 6 *Our strategy achieves a competitive factor of 1.8092.*

Proof. According to Theorem 1, no algorithm can guarantee a better ϕ-value than 0.650245. Our strategy yields an upper bound of 1.1764. This results in a factor of $1.1764/0.650245 \approx 1.8092$. □

5 Discrete Point Sets

Our above analysis relies on continuous weight distributions, which imply the lower bound on ϕ-values stated in Theorem 1. This does include the case of integer values n_i. However, as discussed in [3], considering discrete weight distributions may allow lower average distances; e.g., a single point has an average distance of 0. As a consequence, *towns* (subsets of the integer grid) have lower average distances than cities of the same total weight. However, we still get a competitive ratio for the case of online towns.

Definition 3 ([3]) *An n-town T is a subset of n points in the integer grid. Its* normalized average Manhattan distance *is*

$$\phi(T) = \frac{2c(T)}{n^{5/2}} = \frac{\sum_{s\in T}\sum_{t\in T}\|s-t\|_1}{n^{5/2}}$$

Theorem 7 *For n-towns, our strategy guarantees a competitive factor of at most 1.7849 for the ϕ-value.*

Proof. Analogous to Theorem 5, we consider the values up to $n = 64$, and show that the worst case is attained for $i = 16$, which yields an upper bound of 1.123. (The analog to Table 1 is omitted for lack of space.) For a lower bound, the general value of 0.650245 for ϕ-values cannot be applied, as discrete point sets may have lower average distance. Instead, we verify that the ratio of achieved ϕ to optimal ϕ for $n \leq 64$ is less than 1.7849; for $65 \leq n \leq 80$, Table 1 in [3] allows us to verify that $\phi \geq 0.629171$. For $n \geq 81$, we make use of equation (5), p. 89 of [3] to show the same lower bound of 0.629171; details are slightly involved and omitted for lack of space. Overall, we get a competitive ratio bounded by upper divdided by lower bound, i.e., 1.7849. □

6 Lower Bounds

We demonstrate that there are non-trivial lower bounds for a competitive factor by considering the online scenario for towns.

Theorem 8 *No online strategy can guarantee a competitive factor below $\frac{64}{\sqrt{5}^5} = 1.144866....$*

Proof. Consider a 3x3 square, and let $n_1 = 4$. If the strategy allocates a 2x2 square (for a total distance of 8), then $n_2 = 5$, and the resulting L-shape has a total distance of 20 and a ϕ-value of $40/5^{2.5} = 0.715541...$ Allocating the first town with an L-shape of total distance 10 results in $\phi = 20/32 = 0.625$, and the second with a total distance of 16, or $\phi = 32/5^{2.5} = 0.572433...$

If instead, the first town is allocated different from a square, the total distance is at least 10, and $\phi \geq 20/32$; then $n_2 = n_3 = n_4 = n_5 = n_6 = 1$, and an optimal strategy can allocate the first town as a 2x2 square, with $\phi = 0.5$. This bounds the competitive ratio, as claimed. □

7 Conclusions

The offline problem (where all n_i are given in advance) is interesting in itself. A simple lower bound for a guaranteed maximum is 2/3, as that is the average distance of the whole square. For the case $n_1 = n_2 = 1/2$ we conjecture an optimum of $\sqrt{2}/2$, which we could prove for the special case of divisions using a straight line across the center. We believe the global worst case is attained for $n_1 = n_2 = n_3 = 1/3$.

Our description of the competitive factor of 1.8092 uses the assumption that there is a lower bound ε for the input and a smallest common denominator $c \leq \varepsilon$ for all input numbers. Discarding this assumption is possible, but requires some more tedious analysis.

In principle, further improvement of the established factors could be achieved by replacing the anlysis from $n = 16, \ldots, 64$ by $n = 65, \ldots, 256$. However, the highest known optimal ϕ-values are for $n = 80$ using the $O(n^{7.5})$ algorithm of [3], so the involved computational effort promises to be large.

Acknowledgments

We thank Bettina Speckmann for helpful comments and pointing out reference [10].

References

[1] C. M. Bender, M. A. Bender, E. D. Demaine, and S. P. Fekete. What is the optimal shape of a city? *J. Physics A: Math. Gen.*, 37(1):147–159, 2004.

[2] M. A. Bender, D. P. Bunde, E. D. Demaine, S. P. Fekete, V. J. Leung, H. Meijer, and C. A. Phillips. Communication-Aware Processor Allocation for Supercomputers: Finding Point Sets of Small Average Distance. *Algorithmica*, 50(2):279–298, 2008.

[3] E. D. Demaine, S. P. Fekete, G. Rote, N. Schweer, D. Schymura, and M. Zelke. Integer point sets minimizing average pairwise L1 distance: What is the optimal shape of a town? *Computational Geometry: Theory and Applications*, 40:82–94, 2011.

[4] R. M. Karp, A. C. McKellar, and C. K. Wong. Near-Optimal Solutions to a 2-Dimensional Placement Problem. *SIAM J. Comp.*, 4(3):271–286, 1975.

[5] J. Y.-T. Leung, T. W. Tam, C. S. Wing, G. H. Young, and F. Y. Chin. Packing squares into a square. *J. Parallel Distrib. Comput.*, 10(3):271–275, 1990.

[6] V. J. Leung, E. M. Arkin, M. A. Bender, D. P. Bunde, J. Johnston, A. Lal, J. S. B. Mitchell, C. A. Phillips, and S. S. Seiden. Processor Allocation on Cplant: Achieving General Processor Locality Using One-Dimensional Allocation Strategies. In *Proc. IEEE Int. Conf. Cluster Comp.*, pages 296–304. 2002.

[7] H. Sagan. *Space-Filling Curves*. Springer, 1994.

[8] N. Schweer. *Algorithms for Packing Problems*. PhD thesis, TU Braunschweig, 2010.

[9] R. Siromoney and K. Subramanian. Space-filling curves and infinite graphs. In *Graph-Grammars and Their Application to Computer Science*, volume 153 of *LNCS*, pages 380–391, Berlin, 1983. Springer.

[10] M. Wattenberg. A note on space-filling visualizations and space-filling curves. In *Proc. IEEE Symp. Information Visualization*, pages 181–186, 2005.

Computing Popularity Maps with Graphics Hardware

Marta Fort * J. Antoni Sellarès * Nacho Valladares *

Abstract

The popularity of a point is a measure of how many of a set of moving objects have visited the point. The popularity map is the subdivision of the plane into regions where all points have the same popularity. In this paper we propose an algorithm to efficiently compute popularity maps that takes benefit of the Graphics Processing Unit parallelism capabilities. We also present experimental results obtained with the implementation of our algorithm.

1 Introduction

Mobile devices are able to generate trajectories of moving objects (for example vehicles, people or animals), called entities, available as points representing a position in space in a certain instant of time. Trajectory databases, in many cases rather large in volume, contain valuable and implicit knowledge that needs to be extracted. Several exact and approximate algorithms, based on computational geometry techniques, to detect group movement patterns have been proposed [8, 6, 3, 1, 7].

We are interested in the problem of detecting popular regions among trajectories, they are places that are visited by many entities. The localization of popular regions has multiple applications in real life. In traffic planning, we are interested in the most congested places. In tourism management, we want to know the locations of a historical town which are more visited by tourist. In marketing, we want to ensure the effectiveness of an advertisement in a mall determining how many people have seen it. Depending on the application it is convenient to know the exact number of times that an entity has visited the place or only to know if it has visited the place or not. In the traffic planning example it is necessary to count the number of times that a car has been in a place. In the tourism management example it does not matter whether a tourist has been in the place once or more than once, we are just interested in how many different tourists have been there. In the marketing example, both criteria could be applied.

Popular regions were first studied in [2]. The authors present a continuous model where entire trajectories, described by polylines whose vertices are the positions of entities at consecutive time steps, are taken into account. Given a set T of n trajectories with τ time steps each, $r > 0$ a real value and $k > 0$ an integer, a point p is a (r,k)-popular point if there are at least k different trajectories of T that intersect the square $S(p,r)$. The parameter r models the proximity between the point and the trajectories and parameter k measures the point popularity. Note that the entities do not have to be is the square simultaneously. A (r,k)-popular region is defined as a maximal connected set of (r,k)-popular points. It is not difficult to see that popular regions are polygons. Denote $\mathscr{P}_{r,k}(T)$ the collection of (r,k)-popular regions. In [2] an algorithm, rather difficult to implement, to compute $\mathscr{P}_{r,k}(T)$ that takes $O(\tau^2 n^2)$ time and requires $O(\tau n + V)$ space, where V denotes the total number of vertexes of $\mathscr{P}_{r,k}(T)$, is presented. No results of the implementation are reported. Finally, the paper remarks that another natural way of defining a popular place is by using a disk instead of a square and that more sophisticated techniques are needed to handle this new problem.

In [5] authors presented algorithms, that take benefit of the Graphics Processing Unit (GPU) parallelism capabilities, to detect popular regions in the continuous model when a disk $D(p,r)$ of center p and radius r is used to determine proximity instead of a square. The paper uses a **weak criterion** to count intersections: a trajectory intersecting the disk $D(p,r)$ multiple times counts only once.

In this paper we use a **strong criterion** for counting the number of intersections: the number of intersections between a trajectory t and the disk $D(p,r)$ is the number of maximal sub-trajectories of t contained in $D(p,r)$.

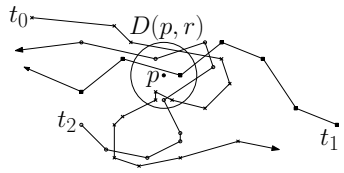


Figure 1: The number of intersections is three for the weak criterion and five for the strong criterion.

The GPU inherent parallelism and the ability to work independently alongside the CPU as a co-processor make it a compelling platform for computationally demanding tasks.

In this paper, by working towards practical solutions, we use Computational Geometry techniques together with the GPU capabilities to detect and visualize popular regions with good running times. Although our proposed approach makes reported solutions approximated, it is acceptable since data of moving entities are also approximated. We also present and discuss experimental results obtained with the implementation of our algorithm.

*Email: {mfort,sellares,ivalladares}@ima.udg.edu. Univ. de Girona, Spain. Partially supported by the Ministerio de Ciencia y Innovación under grant TIN2010-20590-C02-02.

1.1 Graphics pipeline and Cg

The OpenGL graphics pipeline [9] is divided into several stages. The input is a list of 3D geometric primitives expressed as vertexes defining points, lines, etc. with attributes associated. The output is a buffer (also called image) corresponding to a two dimensional $H \times W$ grid whose cells are called pixels, H and W are the height and width screen resolution, respectively. In the first stage of the pipeline, per-vertex operations take place. Each input vertex is transformed from 3D coordinates to window coordinates obtaining 2D primitives. The following stage (rasterization) rasterizes every obtained primitive according to the screen resolution, and fragments, with their attributes, are obtained. Different primitives can be projected in the same 2D space and several fragments can belong to the same pixel position. The last stage, the fragment stage, computes the color, alpha and depth value of each fragment, these values determine the final output. The color of each pixel is obtained by taking into account all fragments corresponding to the given pixel. Finally, the depth and stencil tests, among other tests, take place to determine whether a fragment is painted or not. They are executed in order and only when are enabled. Depth test uses an internal GPU buffer called depth buffer. It is used to discard fragments based on the comparison between the already stored value in the (x,y) buffer position and the incoming one, storing, on its per defect configuration, the closest fragment.

The operations performed in the graphics pipeline conform a sequence of non user controlled consecutive operations. Shader languages like Cg provide the user the ability of modifying some pre-established operations, to adapt the pipeline to the user needs by using 'shaders', small programs which modify the stage behavior.

2 Popularity maps

Let T be a set of n trajectories $T = \{t_0, \ldots, t_{n-1}\}$. Each trajectory t_i is a sequence of τ points in the plane, $t_i : p^i_0, \ldots, p^i_{\tau-1}$, where p^i_j denotes the position of entity e_i at time j with $0 \leq j \leq \tau - 1$. We assume a continuous model in which the movement of an entity e_i from its position p^i_j to its position p^i_{j+1} is described by the straight-line segment joining p^i_j and p^i_{j+1}, this movement is supposed to be done in a constant speed. The trajectory t_i is described by the polyline, which may self-intersect, whose vertices are the trajectory points $t_i : p^i_0, \ldots, p^i_{\tau-1}$.

For a given parameter $r > 0$, the **popularity of a point** p is the total number of intersections between the trajectories of T and the disk $D(p,r)$ of center p and radius r. The **popularity map**, $\mathscr{M}_r(T)$, is the partition of the plane in maximal connected regions so that all points of a region have the same popularity. Notice that the set of points of a given popularity may be composed of several independent connected regions bounded by straight-line segments and circular arcs.

For each edge $e^i_j = p^i_j p^i_{j+1}$ of the polyline representing trajectory t_i we consider the offset region $O_r(e^i_j)$ obtained sweeping along e^i_j a disk of radius r such that the center moves on e^i_j. Then we define $P_r(e^i_j)$ as $P_r(e^i_j) = O_r(e^i_j)$ if $j = 0$ and $P_r(e^i_j) = O_r(e^i_j) \setminus D(p^i_j, r)$ otherwise. The popularity map $\mathscr{M}_r(T)$ can be obtained from the arrangement of regions $P_r(e^i_j)$, $0 \leq j \leq \tau - 1$, $0 \leq i \leq n-1$ (Figure 2.a). Notice that the intersection between the regions $P_r(e^i_j)$ and $P_r(e^i_{j+1})$ determined by two consecutive edges e^i_j, e^i_{j+1} of trajectory t_i, defines a region such that a disk centered in any of their points and radius r intersects t_i in two maximal subtrajectories, one contained in e^i_j and the other in e^i_{j+1}, and therefore the number of intersections between t_i and the disk is two.

To compute $\mathscr{M}_r(T)$ we will discretize the plane into $H \times W$ points such that each point corresponds to a pixel in the graphics pipeline. The idea is to send regions $P_r(e^i_j)$ to the GPU as a set of rectangles and disks. Inside the pipeline all primitives will be rasterized into fragments according to the screen resolution $H \times W$. Then, we will store the number of fragments corresponding to each pixel. In this way we obtain the popularity of each pixel and consequently a popularity map discretization $\mathscr{M}_r(T)$ (Figure 2.b).

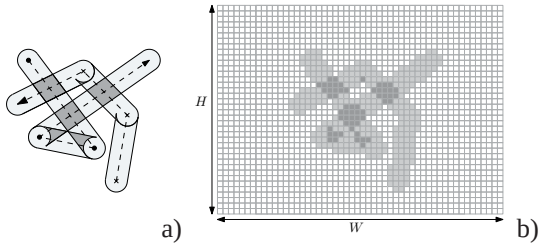


Figure 2: a) Popularity map. b) Discretized popularity map.

The disk $D(p^i_j, r)$ is painted as a square centered at p^i_j of side $2r$. Then a fragment shader is activated such that fragments whose Euclidean distance to p^i_j is bigger than r are discarded.

2.1 Computing popularity maps

Given a value of parameter r, we could compute the popularity map under the strong criterion with a similar algorithm to the one used in [5]. There, each trajectory is painted at a different depth and the stencil buffer is used to count the number of fragments with different depth that correspond to each pixel. To handle the strong criterion we should paint each trajectory edge, instead of the whole trajectory, at a different depth. To avoid overflows, the stencil buffer has to be read to CPU every 255 time steps which is too much often. Thus we propose an alternative method which avoids this read backs to CPU.

The idea is to paint each $P_r(e^i_j)$ with a different depth value and via fragment shader increment, for each fragment, the color value of its pixels. When all $P_r(e^i_j)$ are painted we can determine how many fragments correspond

to each pixel depending on its final color.

The algorithm needs to add color to each pixel for each fragment. Since we can not read from a texture and at the same time write on it, we use two textures. Texture A with the previous color of each pixel, and texture B to store the new accumulated color. At each rendering pass the accumulated colors of B are copied to texture A to be recovered for the next time step as the last color stored.

Since we have 3 channels of color (RGB) with 8 bits per channel $[0\ldots255]$ we add 1 to the first channel for each fragment until the red channel is overflowed. Then we increment the green channel with 1 and we set the red channel to 0, adding again to red channel until it is overflowed again. The same process is applied to the blue channel when the green channel is overflowed. We can store up to 2^{24} values which is more than enough in most applications.

The algorithm proceeds as follows. We paint each offset region $O_r(e^i_j)$ at a different depth value $z = z^i_j$ from further ($z = 1$) to closer ($z = 0$) (Figure 3). The fragment shader is activated during all the process not just for adding the color values for all fragments but also to paint a disk when needed as explained in Section 2. Since we have a depth precision of 32 bits we can paint up to 2^{32} time steps at different depth levels.

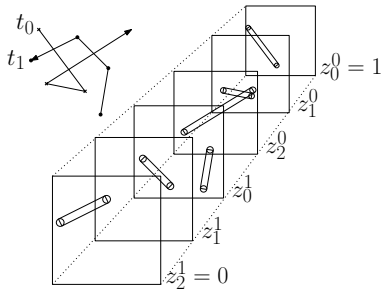


Figure 3: Offset regions rendered from z=1 to z=0.

Notice that the disk corresponding to p^i_j, $1 \le j \le \tau - 2$, is painted twice, one for each trajectory edge it defines, with different depth values. This causes the color value to be incremented twice where it should count only once. In addition the overlapped regions between rectangles and disks must be avoided too. To solve this problem we add a parameter to the shader to inform of whether the disk fragments have to increment the color or only modify the depth value. The shader updates the depth value but does not increment the color when painting the first disk. When the rectangle and the second disk are painted, both, color and depth values are updated. This way the overlapped fragments between the two disks and the rectangle will be discarded by the depth test and the disks painted twice are counted just once.

While the number of time steps is smaller than 2^{32} and the popularity of a point is smaller than 2^{24} not read backs to CPU have to be done. This turns to 0 the number of read back from GPU in most applications.

2.2 Complexity analysis

We will use the following notation. We denote by P_x the time needed to render and color x fragments, A_x the time spent to make x accesses to a texture, C_x the one needed to copy x pixels from texture to texture.

The number of painted disks for each trajectory is $O(2\tau)$, from which, τ make accesses to texture values to update the color. They represent $8\tau r^2 nHW$ painted pixels and $\tau 4\pi r^2 HW$ texture accesses. Concerning the rectangles, both, the number of accesses to a texture and pixels painted is $2LrHW$, where L is the sum of all the trajectories length. Finally the information of texture B is copied to texture A a total of τ times per trajectory, providing $HW\tau n$ copied values. Thus, the time complexity is $O(P_{(8\tau r^2 n+2Lr)HW} + A_{(n4\pi r^2\tau+2Lr)HW} + C_{n\tau HW})$ and no extra space is needed.

3 Popular regions

Let T be a set of n trajectories, $r > 0$ be a real value and $k > 0$ an integer. A point p is a (r,k)-**popular point** if the total number of intersections between the disk $D(p,r)$ of center p and radius r and the trajectories of T is at least k. We define a (r,k)-popular region as a maximal connected set of (r,k)-popular points. A (r,k)-popular region is bounded by straight-line segments and circular arcs. It is not difficult to see that a (r,k)-popular region is a maximal connected region conformed by regions of the popularity map $\mathscr{M}_r(T)$ whose points have popularity at least k. We denote $\mathscr{P}_{r,k}(T)$ the collection of (r,k)-popular regions (Figure 4).

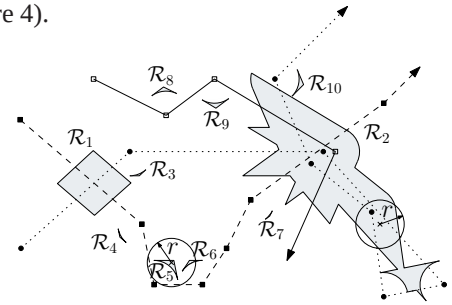


Figure 4: Example with 3 trajectories. Points marked with crosses are $(r,2)$-popular points. We have $\mathscr{P}_{r,2}(T) = \{\mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_{10}\}$.

A discretization of $\mathscr{P}_{r,k}(T)$ can be easily obtained from the discretized $\mathscr{M}_r(T)$. We assign to a pixel of $\mathscr{P}_{r,k}(T)$ value 1 if its corresponding pixel in $\mathscr{M}_r(T)$ has popularity at least k and value 0 otherwise.

3.1 Popular regions visualization

We could visualize $\mathscr{P}_{r,k}(T)$ as a binary image painting in black, pixels whose $\mathscr{P}_{r,k}(T)$ value is 0 an in white those with value 1. Alternatively, in order to obtain a better visual information, we can visualize $\mathscr{P}_{r,k}(T)$ from the discretized $\mathscr{M}_r(T)$: pixels of $\mathscr{M}_r(T)$ with value less than k

are painted white and the rest of pixels are painted according to its popularity. Then, instead of having just two colors, we uniformly distribute the popularity range values among the whole RGB range (red, green and blue) (See Figure 5.b). In particular, when $k = 1$ we obtain the visualization of the popularity map $\mathscr{M}_r(T)$.

4 Results

Tests have been computed in a Intel Core 2 CPU 2.13GHz, 2GB RAM and a GPU NVidia GeForce GTX 480. Each running time reported in Table 1 is the average of 10 executions with the same parametrization.

The algorithm has been tested under different data sets. 'Animals Sim.' is a synthetic data sets generated with Netlogo [11] where 10.000 animals move on a terrain with no movement restrictions interacting each other to get closer. A total of 200,000 time steps are tracked. 'Buses' is a set of school buses moving in Athens metropolitan area extracted from [10] with 145 buses registered during 66,096 time steps. Finally 'State Fair' is a daily GPS track collected from the human movement in NC State Fair held in North Carolina [4], 19 persons are tracked with a total of 5,861 time steps. Table 1 shows the running times needed to compute $\mathscr{M}_r(T)$ plus $\mathscr{P}_{r,k}(T)$ and to visualize $\mathscr{P}_{r,k}(T)$ at resolutions 512×512 and 1024×1024.

			Computation (s)		*Visualization (ms)*	
T	r	k	512^2	1024^2	512^2	1024^2
State fair	2	5	0.344	0.440	3.836	12.609
	5	5	0.351	0.439	3.293	10.564
	10	5	0.346	0.437	2.604	8.764
	5	2	0.346	0.437	2.506	8.938
	5	10	0.347	0.437	4.053	12.589
	5	50	0.346	0.450	6.445	21.304
Buses	2	5	3.401	4.469	2.507	9.241
	5	5	3.399	4.470	2.315	8.864
	10	5	3.400	4.408	2.249	8.716
	5	2	3.402	4.465	2.217	8.588
	5	10	3.398	4.474	2.457	9.179
	5	50	3.401	4.585	3.078	10.661
Animals Sim.	2	5	10.148	12.808	12.824	57.552
	5	5	10.164	12.812	12.744	55.121
	10	5	10.168	12.805	12.435	50.165
	5	2	10.114	12.788	12.126	47.694
	5	10	10.166	12.755	12.842	57.822
	5	50	10.125	13.138	12.692	58.314

Table 1: Computational (in seconds) and visualization (in milliseconds) times for different data sets of trajectories.

From the table we conclude that the computation and visualization times are fairly affected by r or k. Note that the bigger the number of time steps, the bigger the running times. This is what we expect since the number of renders and fragment processing is directly proportional to the number of time steps. The provided inputs, from 5,861 to 200,000 time steps, show a good scalability of our algorithm. We can not compare our execution times against others because, from the best of our knowledge, no other implementations exist.

It is easy to see that the error produced by our algorithm due to space discretization into pixels is inversely proportional to the discretization size and directly proportional to the covered area of the input data. For instance if the input data covers a squared area of 16 Km^2 the error produced is of 3.9 meters per pixel. This is a reasonable error because GPS signals also produce similar errors. To decrease the error we can increment the discretization size, this has GPU hardware limitations. A possible solution is to subdivide the area into sub-areas and apply the algorithm for each one increasing the discretization size.

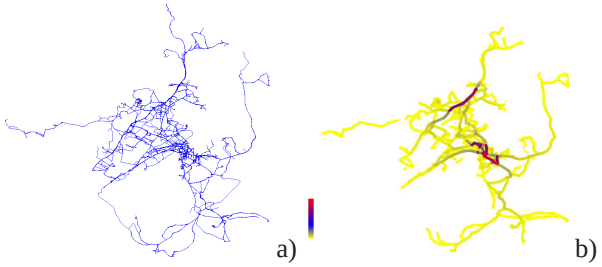


Figure 5: a) Buses dataset. b) $\mathscr{P}_{r,k}(T)$ visualization. Yellow, grey/blue and red regions means low, median and high popularity respectively.

References

[1] M. Andersson, J. Gudmundsson, P. Laube, and T. Wolle. Reporting Leaders and Followers among Trajectories of Moving Point Objects. *GeoInformatica*, 12(4):497–528, 2008.

[2] M. Benkert, B. Djordjevic, J. Gudmundsson, and T. Wolle. Finding popular places. *Int. Journal of Computational Geometry and Applications (IJCGA)*, 20(1):19–42, 2010.

[3] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008.

[4] C. Dartmouth. CRAWDAD. `http://crawdad.cs.dartmouth.edu/index.php`.

[5] M. Fort, J. A. Sellarès, and N. Valladres. Computing popular places using graphics processors. In *Proc. SSTDM-10 in cooperation with IEEE ICDM-10*, pp 233-240, 2010.

[6] J. Gudmundsson, M. J. van Kreveld, and B. Speckmann. Efficient Detection of Patterns in 2D Trajectories of Moving Points. *GeoInformatica*, 11(2):195–215, 2007.

[7] J. Gudmundsson, and M. J. van Kreveld. Computing longest duration flocks in trajectory data. In R. A. de By and S. Nittel, editors, *GIS*, pages 35–42. ACM, 2006.

[8] P. Laube, M. van Kreveld, and S. Imfield. Finding REMO - Detecting Relative Motion Patterns in Geospatial Lifelines. *Developments in Spatial Data Handling: 11th Int. Sympos. on Spatial Data Handling*, pp 201–215, 2004.

[9] M. Segal and K. Akeley. The design of the opengl graphics interface. Technical report, Silicon Graphics Computer Systems, 1994.

[10] Y. Theodoridis. R-Tree portal. `http://www.rtreeportal.org`.

[11] U. Wilensky. NetLogo. `http://ccl.northwestern.edu/netlogo`.

Wireless Localization with Vertex Guards

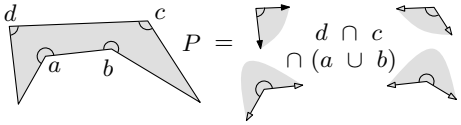
Aurosish Mishra* Tobias Christ†

Abstract

We consider the *wireless localization problem*. Given a simple polygon P, place and orient guards each of which broadcasts a unique key within a fixed angular range. At any point in the plane one must be able to tell whether or not one is located inside P only by looking at the set of keys received. In other words, the interior of the polygon must be described by a monotone Boolean formula composed from the guards. We improve the upper bound for the vertex guard problem where guards may be placed on vertices of P only and show that the maximum number of vertex guards needed to describe any simple polygon on n vertices is at most $\frac{8}{9}n$.

1 Wireless Localization

Art gallery problems are a classic topic in discrete and computational geometry. A new direction has been introduced by Eppstein, Goodrich, and Sitchinava [4]. They propose to modify the concept of visibility by not considering the edges of the polygon/gallery as blocking. This changes the problem drastically. The motivation for this model stems from communication in wireless networks where the signals are not blocked by walls, either. For illustration, suppose you run a café (modeled as a simple polygon P) and you want to provide wireless internet access. But you do not want the whole neighborhood to use your infrastructure. Instead, Internet access should be limited to those people who are located within the café. To achieve this, you can install a certain number of devices, let us call them guards, each of which broadcasts a unique (secret) key in an arbitrary but fixed angular range. The goal is to place guards and adjust their angles in such a way that everybody who is inside the café can prove this fact just by naming the keys received and nobody who is outside the café can provide such a proof. Formally this means that P must be described by a monotone Boolean formula over the keys, that is, a formula using the operators AND and OR only, negation is not allowed. It is convenient to model a guard as a subset of the plane, namely the area where the broadcast from this guard can be received. This



area can be described as an intersection or union of at most two halfplanes. Using this notation, the polygon P is to be described by a combination of the operations union and intersection over the guards. For example, the polygon P above can be described by $(a \cup b) \cap c \cap d$.

Natural guards. Natural locations for guards are the vertices and edges of P. A guard which is placed at a vertex of P is called a *vertex guard*. A vertex guard is *natural* if it covers exactly the interior angle of its vertex. Natural vertex guards alone do not always suffice [4]. A guard placed anywhere on the line given by an edge of P and broadcasting within an angle of π to the inner side of the edge is called a *natural edge guard*. Dobkin, Guibas, Hershberger, and Snoeyink [3] showed that n natural edge guards are sufficient for any simple polygon with n edges. Using both natural vertex guards and natural edge guards, $n-2$ guards are sufficient and can be necessary [1].

Vertex guards. Using a different approach, Eppstein et al. [4] proved that any simple polygon with n edges can be guarded using at most $n-2$ (general, that is, not necessarily natural) vertex guards. In this work, we improve the upper bound to $\lfloor \frac{8n-6}{9} \rfloor$ for $n \geq 4$. This bound is still not known to be tight. Damian, Flatland, O'Rourke, and Ramaswami [2] describe a family of simple polygons with n edges which require at least $\lfloor 2n/3 \rfloor - 1$ vertex guards.

General guards. In the most general setting, we do not have any restriction on the placement and the angles of guards. At the moment, the best known upper bound is $\lfloor \frac{4n-2}{5} \rfloor$, which is not known to be tight, the best lower bound being $\lceil \frac{3n-4}{5} \rceil$ [1].

The different problems and results are summarized in the following table. The mark $*$ indicates the result of this paper.

guards	lower bound	upper bound
natural	$n-2$ [1]	$n-2$ [3]
vertex	$\lfloor 2n/3 \rfloor - 1$ [2]	$\lfloor (8n-6)/9 \rfloor$ [$*$]
general	$\lceil (3n-4)/5 \rceil$ [1]	$\lfloor (4n-2)/5 \rfloor$ [1]

*Dept. of Computer Science and Engineering, IIT Kharagpur, aurosish007@gmail.com

†Institute of Theoretical Computer Science, ETH Zurich tobias.christ@inf.ethz.ch

2 Upper Bound for Vertex Guards

We use the notion of a *polygonal halfplane* H which is a topological halfplane bounded by a *simple bi-infinite polygonal chain* $C = (e_1, \ldots, e_n)$, for a positive integer n. For $n = 1$, the only edge e_1 is a line and the polygonal halfplane is a halfplane. For $n = 2$, e_1 and e_2 are rays which share a common source but are not collinear. For $n \geq 3$, e_1 and e_n are rays, e_i is a line segment, for $1 < i < n$, and e_i and e_j, for $1 \leq i < j \leq n$, do not intersect unless $j = i+1$ in which case they share an endpoint. For brevity we use the term *chain* in place of simple bi-infinite polygonal chain. Let v_i, for $1 \leq i < n$, denote the vertex of C incident to e_i and e_{i+1}, $V(C) := \{v_1, \ldots, v_{n-1}\}$. For $2 \leq i \leq n-1$, let e_i^+ be the ray obtained from e_i by extending the segment linearly beyond v_i. Similarly e_i^- refers to the ray obtained from e_i by extending the segment linearly beyond v_{i-1}. For a polygonal halfplane H define $\gamma(H)$ to be the minimum integer k such that there exists a guarding $\mathcal{G}(H)$ for H using k vertex guards. Similarly, for a natural number n, denote by $\gamma(n)$ the maximum number $\gamma(H)$ such that H is bounded by a chain with n edges. Obviously, $\gamma(1) = \gamma(2) = 1$. Observe that any guarding for H can be transformed into a guarding for the complement $\overline{H}$ using the same number of guards: Use de Morgan's rules and invert all guards (keep their location but flip the angle to the complement with respect to 2π). Therefore, we can define $\gamma(C) = \gamma(H) = \gamma(\overline{H})$.

Theorem 1 *For any $n \geq 2$, $\gamma(n) \leq \lfloor \frac{8n-3}{9} \rfloor$.*

Proof. The base cases $\gamma(1) = \gamma(2) = 1$, $\gamma(3) = 2$, $\gamma(4) = 3$, $\gamma(5) = 4$ and $\gamma(6) = 5$ follow from the observation that a chain can always be guarded with $n-1$ natural guards [1]. Now let H be a polygonal halfplane bounded by an oriented polygonal chain C with $n \geq 7$ edges such that the interior of H lies to the right of C. Let $S := V(\mathrm{conv}(V(C)))$ be the vertices of the convex hull of $V(C)$, that is, the vertices of C that are extremal. The basic idea is to *split* C at a vertex $v_i \in S$ into two chains $C_1 = (e_1, \ldots, e_{i-1}, e_i^+)$ and $C_2 = (e_{i+1}^-, e_{i+2}, \ldots, e_n)$. If the "new" rays e_i^+ and e_{i+1}^- do not intersect the "old" rays e_1 and e_n, we can express H as the intersection or union of the two polygonal halfplanes H_1 and H_2 bounded by C_1 and C_2 depending on whether v_i is convex or reflex. Assume that the angle between e_1 and e_n is convex (else, consider $\overline{H}$ instead of H) and think of C as going from the left to the right (thus H being below C). In other words, e_1 and e_n are assumed to go from left to right, e_1 having positive slope and e_n having smaller slope than e_1. Now look at the convex hull $\mathrm{conv}(H)$ of H. There must be at least one vertex v_i in S which lies on the boundary $\partial\mathrm{conv}(H)$. Such a vertex is for sure a good splitting vertex in the above sense. If $2 \leq i \leq n-2$, we split C at v_i as explained into two chains $C_1 := (e_1, \ldots, e_{i-1}, e_i^+)$ and $C_2 := (e_{i+1}^-, e_{i+2}, \ldots, e_n)$ and get a guarding $\mathcal{G}(C_1) \cap \mathcal{G}(C_2)$, where $\mathcal{G}(C_i)$ denotes the guarding of C_i we get by induction, see Figure 1. Therefore $\gamma(C) \leq \gamma(i) + \gamma(n-i) \leq \lfloor \frac{8i-3}{9} \rfloor + \lfloor \frac{8(n-i)-3}{9} \rfloor \leq \lfloor \frac{8n-6}{9} \rfloor \leq \lfloor \frac{8n-3}{9} \rfloor$.

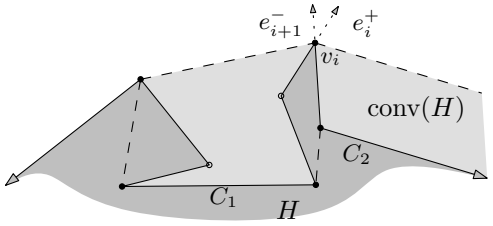


Figure 1: Splitting at a convex hull vertex.

If there is no vertex on $\partial\mathrm{conv}(H)$ with index $2 \leq i \leq n-2$, we first consider the case that $S \cap \partial\mathrm{conv}(H) = \{v_1, v_{n-1}\}$. If there is a vertex $v_i \in S$ with $3 \leq i \leq n-3$, split C at v_1, v_i and v_{n-1}: Put a natural edge guard g_1 onto e_1, a natural edge guard g_2 onto e_n and define $C_1 := (e_2^-, \ldots, e_i^+)$, $C_2 := (e_{i+1}^-, \ldots, e_{n-1}^+)$. (We can place the natural edge guards on the incident vertices, hence the natural edge guards can be realized as (non-natural) vertex guards.) Then, a guarding for C can be obtained as $g_1 \cap g_2 \cap (\mathcal{G}(C_1) \cup \mathcal{G}(C_2))$, see Figure 2. This implies $\gamma(C) \leq 2 + \gamma(i-1) + \gamma(n-i-1) \leq 2 + \lfloor \frac{8i-11}{9} \rfloor + \lfloor \frac{8(n-i)-11}{9} \rfloor \leq \lfloor \frac{8n-6}{9} \rfloor$.

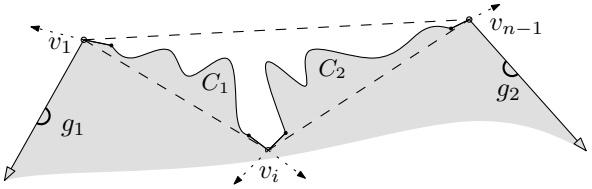


Figure 2: v_1 and v_{n-1} are the only vertices on $h(C)$.

If $S = \{v_1, v_2, v_{n-2}, v_{n-1}\}$, consider $S' := V(\mathrm{conv}\{v_2, \ldots, v_{n-2}\})$. Beside v_2 and v_{n-2}, there must be a third vertex $v_j \in S'$, without loss of generality $4 \leq j \leq n-3$ (if $j = 3$, reflect C). If e_3^- does not intersect e_n, put a natural edge guard g_1 onto e_1, a vertex guard g_2 onto v_2 with the right ray covering e_2 and its other ray parallel to e_n and a natural vertex guard g_3 onto v_{n-1} and define $C_1 := (e_3^-, \ldots, e_j^+)$ and $C_2 := (e_{j+1}^-, e_{j+2}, \ldots, e_{n-2}, r)$ where r is the ray starting at v_{n-2} in the direction of e_n. See Figure 3. Then we get a guarding as $g_1 \cap (g_2 \cup (\mathcal{G}(C_1) \cap \mathcal{G}(C_2)) \cup g_3)$ and conclude $\gamma(C) \leq 3 + \gamma(j-2) + \gamma(n-j-1) \leq 3 + \lfloor \frac{8(j-2)-3}{9} \rfloor + \lfloor \frac{8(n-j-1)-3}{9} \rfloor \leq \lfloor \frac{8n-3}{9} \rfloor$. If e_3^- intersects e_n, put a natural vertex guard g_1 onto v_1, and an edge guard g_2 onto e_n and define $C' := (e_3^-, \ldots, e_{n-1}^+)$. See Figure 3. We obtain a guarding $(g_1 \cup \mathcal{G}(C')) \cap g_2$.

If S consists of 3 vertices only and there is no $v_i \in S$ with $3 \leq i \leq n-3$, assume without loss of generality that $v_2 \in S$ (if v_{n-2} is the only vertex in S beside

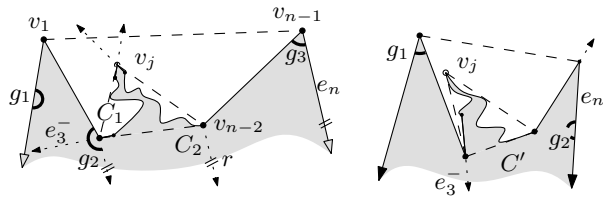


Figure 3: $S = \{v_1, v_2, v_{n-2}, v_{n-1}\}$

v_1 and v_{n-1}, reflect C). In this case, define $S' := V(\text{conv}(\{v_2, \ldots, v_{n-1}\}))$. For sure, $v_2, v_{n-1} \in S'$ but there must be a third vertex $v_j \in S'$, see Figure 4.

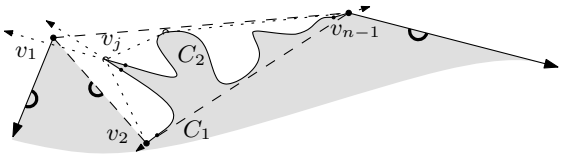


Figure 4: v_1, v_2, and v_{n-1} are the only vertices in S.

If $4 \leq j \leq n-3$, put a edge guards onto e_1, e_2, and e_n, and split the remaining chain at v_j. If v_3 is the only new vertex in S', put a natural vertex guard g_1 onto v_1, and a non-natural vertex guard g_2 onto v_2 covering e_3 with its left ray and with the right ray parallel to e_1, and an edge guard g_3 onto e_n: A guarding can be obtained as $g_3 \cap (g_1 \cup (g_2 \cap \mathcal{G}(C')))$ where $C' = (e_4^-, \ldots, e_{n-1}^+)$. See Figure 5.

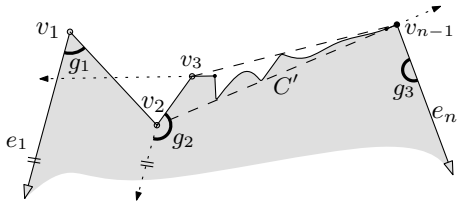


Figure 5: $S' = \{v_{n-1}, v_3, v_2\}$.

If v_{n-2} is the only new vertex, define $C' = (e_3^-, \ldots, e_{n-2}^+)$, put a natural vertex guard g_3 onto v_{n-1}. If e_3^- does not intersects e_n, put an edge guard g_1 onto e_1 and a vertex guard g_2 onto v_2 covering e_2 with its right ray and with its left ray parallel to e_n. We get a guarding $g_1 \cap (g_2 \cup (\mathcal{G}(C') \cap g_3))$. If e_3^- intersects e_n, put a natural vertex guard g_1 onto v_1 and an edge guard g_2 onto e_n and observe $H = g_2 \cap (g_1 \cup (\mathcal{G}(C') \cap g_3))$. We conclude $\gamma(C) \leq 3 + \gamma(n-4) \leq 3 + \lfloor \frac{8n-32-3}{9} \rfloor \leq \lfloor \frac{8n-8}{9} \rfloor$, see Figure 6.

Finally, assume there is only one vertex on $\partial\text{conv}(H)$, which is either v_1 or v_{n-1}. We assume without loss of generality that it is v_1. Beside v_1, which for sure belongs to S, there must be at least two more vertices in S. Let v_i be the vertex of S which is extremal to the right of e_n. If $3 \leq i \leq n-2$, split C into three parts cutting it at v_1 and v_i. Then, we get a guarding for C as $g \cap (\mathcal{G}(C_1) \cup \mathcal{G}(C_2))$, where

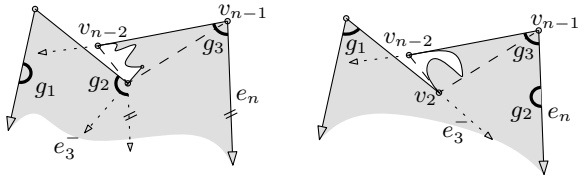


Figure 6: $S' = \{v_{n-1}, v_{n-2}, v_2\}$.

g is a natural edge guard on e_1, $C_1 = (e_2^-, \ldots, e_i^+)$, and $C_2 = (e_{i+1}-, \ldots, e_n)$, see Figure 7.

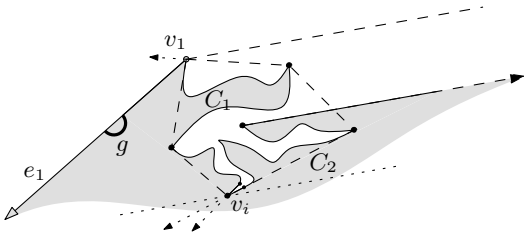


Figure 7: Split at the extremal vertex below e_n.

If $i = 2$, define $S' := V(\text{conv}(\{v_2, \ldots, v_{n-1}\}))$. Let v_j be the vertex of S' which is extremal in the opposite direction, that is, to the left of e_n. If $4 \leq j \leq n-2$, we put natural edge guards g_1 and g_2 onto e_1 and e_2 and split the rest at v_j into two chains C_1 and C_2, see Figure 8. Then $H = g_1 \cap (g_2 \cup (\mathcal{G}(C_1) \cap \mathcal{G}(C_2)))$.

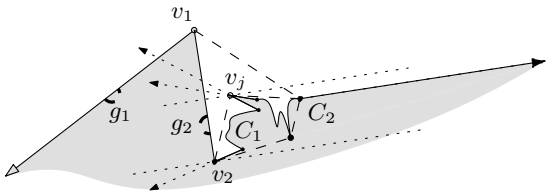


Figure 8: The extremal vertex in S below e_n is v_2.

If $j = 3$, we put a natural vertex guard onto v_1, a guard onto v_2 with its left ray covering e_3 and the right parallel to e_1. Then, we get $\gamma(C) \leq 2 + \gamma(n-3) \leq 2 + \lfloor \frac{8n-27}{9} \rfloor \leq \lfloor \frac{8n-9}{9} \rfloor$. If $j = n-1$, that is, if there is no vertex above e_n except v_1, take any vertex $v_s \in S'$, $3 \leq s \leq n-2$. If $4 \leq s \leq n-3$, put an edge guard g_1 onto e_1 and an edge guard g_2 onto e_2 and an edge guard g_3 onto e_n and define $C_1 = (e_3^-, \ldots, e_s^+)$, $C_2 := (e_{s+1}^-, \ldots, e_{n-1})$, then we get a guarding $g_1 \cap (g_2 \cup (g_3 \cap (\mathcal{G}(C_1) \cup \mathcal{G}(C_2))))$ (or $g_1 \cap (g_2 \cup (g_3 \cap \mathcal{G}(C_1) \cap \mathcal{G}(C_2)))$ if v_s is convex), see Figure 9. If $s = 3$, put 3 guards explicitly depending on whether v_3 is reflex or convex and cover the remaining chain $C' = (e_4^-, \ldots, e_{n-1}^+)$ recursively, see Figure 10. If $s = n-2$, we are in a situation similar to one of those shown in Figure 3 or 6 and proceed accordingly.

If $i = n-1$, that is, if there is no vertex of S below e_n, distinguish two cases: Either there is a convex vertex in S between v_{n-1} and v_1, or there is no such vertex. If there is a convex vertex $v_j \in S$, with

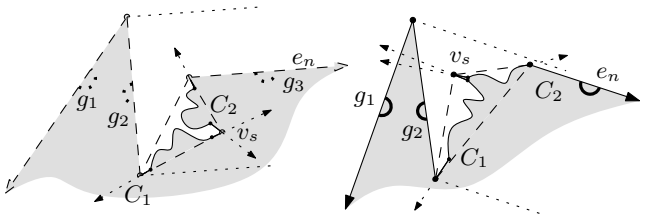


Figure 9: There is no vertex in S' above e_n.

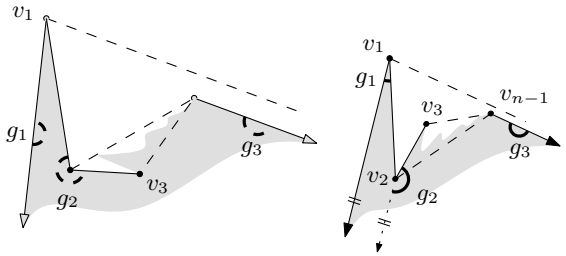


Figure 10: $S' = \{v_2, v_3, v_{n-1}\}$.

$2 \leq j \leq n-3$, put an edge guard onto e_n and split at v_j, see Figure 11. If $j = 2$ or $j = n-1$ proceed as above but now removing v_1 or v_{n-1}, respectively, defining $S' := V(\text{conv}(\{v_2, \ldots, v_{n-1}\}))$ ($S' := V(\text{conv}(\{v_1, \ldots, v_{n-3}\}))$, respectively), see Figure 12.

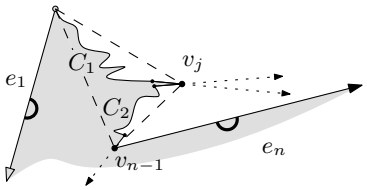


Figure 11: There is a convex vertex $v_j \in S$.

If $S = \{v_1, v_2, v_{n-1}\}$ and v_2 is reflex, let $S' = V(\text{conv}\{v_2, ..., v_{n-1}\})$ and look for a splitting vertex in S'. If there is $v_j \in S'$, $4 \leq j \leq n-3$, put edge guards onto e_1, e_2 and e_n and split the rest at j, see Figure 13. If $S' = \{v_2, v_{n-1}, v_3\}$ or $S' = \{v_2, v_{n-1}, v_3\}$, proceed as shown in Figure 14. □

Lemma 2 (Lemma 4 in [1]) *A simple polygon P on $n \geq 4$ vertices is the intersection of two polygonal halfplanes both having at least two edges.*

Corollary 3 *A simple polygon P on $n \geq 4$ edges can be guarded with at most $\lfloor (8n-6)/9 \rfloor$ vertex guards.*

References

[1] T. Christ, M. Hoffmann, Y. Okamoto, and T. Uno. Improved bounds for wireless localization. *Algorithmica*, 57:499–516, July 2010.

[2] M. Damian, R. Flatland, J. O'Rourke, and S. Ramaswami. A new lower bound on guard placement for wireless localization. `http://arxiv.org/pdf/0709.3554v1`.

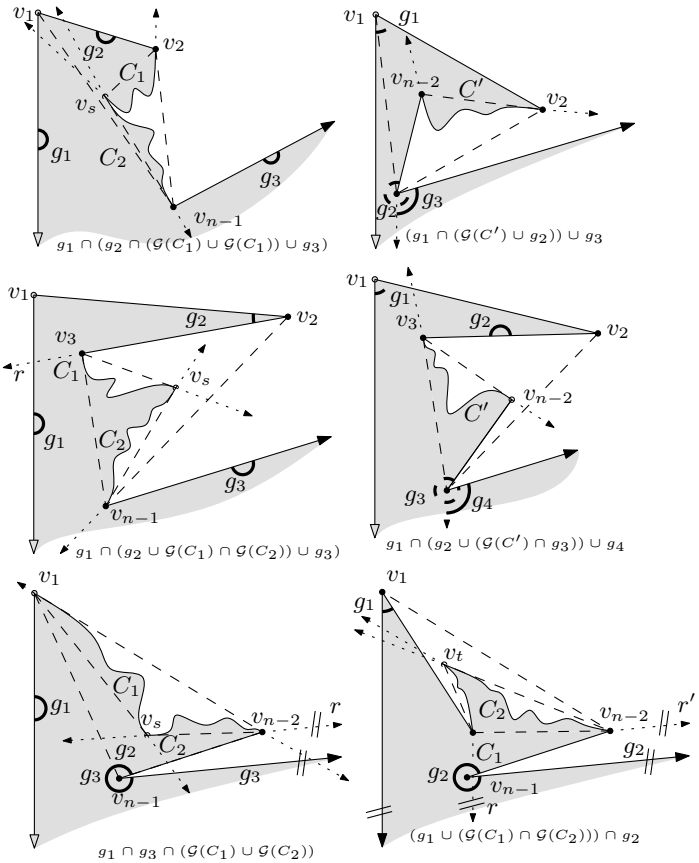


Figure 12: $S = \{v_1, v_2, v_{n-1}\}$ or $S = \{v_1, v_{n-2}, v_{n-1}\}$.

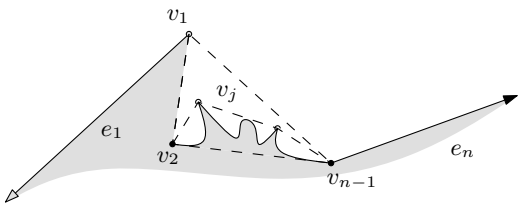


Figure 13: $S = \{v_1, v_2, v_{n-1}\}$.

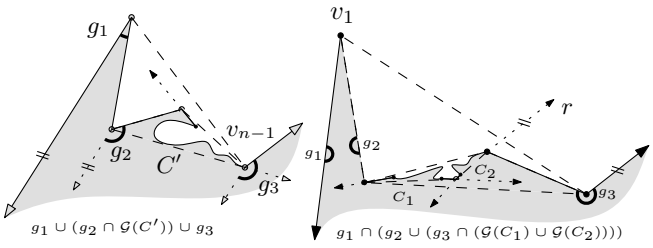


Figure 14: The only new vertex is v_3 or v_{n-1}.

[3] D. P. Dobkin, L. Guibas, J. Hershberger, and J. Snoeyink. An efficient algorithm for finding the CSG representation of a simple polygon. *Algorithmica*, 10:1–23, 1993.

[4] D. Eppstein, M. T. Goodrich, and N. Sitchinava. Guard placement for efficient point-in-polygon proofs. In *Proc. 23rd Annu. Sympos. Comput. Geom.*, pages 27–36, 2007.

Exact medial axis of quadratic NURBS curves

George M. Tzoumas*

Abstract

We study the problem of the exact computation of the medial axis of planar shapes the boundary of which is defined by piecewise conic arcs. The algorithm used is a tracing algorithm, similar to existing numeric algorithms. We trace the medial axis edge by edge. Instead of keeping track of points on the medial axis, we are keeping track of the corresponding footpoints on the boundary curves, thus dealing with bisector curves in parametric space. We exploit some algebraic and geometric properties of the bisector curves that allow for efficient trimming and we represent bifurcation points via their associated footpoints on the boundary, as algebraic numbers. The algorithm computes the correct topology of the medial axis identifying bifurcation points of arbitrary degree.

1 Introduction

The *medial axis* (MA) of an object can be defined as the locus of the centres of maximal bitangent disks and was originally introduced by [1]. The medial axis of a simple closed shape has a tree-like structure that provides an efficient shape representation. Along with a radius function, the MA allows for restoration of the original object, an operation called *medial axis transform* (MAT).

Due to its wide range of applications in biology, path planning and pattern analysis it has been studied extensively [4, 3, 11]. However most efforts consist of numerical algorithms to trace the bisector curves. Another approach is to approximate the boundary of the shape with simpler curves or line segments. However the resulting MA may require expensive post-processing, or be incorrect, as it is very sensitive to perturbations of the boundary of the shape.

Recently, with the availability of algebraic libraries and faster CPUs, there has been interest in exact algorithms following the exact computation paradigm. Successful examples of this approach are the exact and efficient arrangement of conic arcs and the Voronoi diagrams of circles and ellipses [7].

Following this trend, we apply exact computation techniques to the problem of medial axis computation. In theory, one can study some algebraic system and map the problem to the study of an arrangement of algebraic curves, parts of which are the medial axis edges. However, this can lead to highly inefficient computations due to the degree and bitsize explosion of the computed quantities. Therefore, we focus on the parametric representation of the shape, trying to understand the deeper relation between simple geometric constructions (like the tangent line and the circle of curvature) and the parametric bisector curves.

We chose to use a simple $O(N^2)$ algorithm that also works with objects with holes. The algorithm includes a tracing step which we substitute with an exact one. Therefore, no tracing is included and all critical points are guaranteed to be identified. The idea is to start tracing the medial axis, always computing the next branching or terminal point in one step. The algorithm used is adapted from [12]. The branching point is the center of a maximal disk that is tangent to at least three distinct footpoints, while a terminal point corresponds to a local curvarture maximum ([4]). We work in the parametric space, in a manner similar to [13], however for efficient trimming of the bisector curves we exploit some algebraic and geometric properties.

2 Representation

The input is a simply connected closed shape consisting of a sequence of NURBS curves of degree 2, that is, conic arcs. The shape may contain holes, that are also piecewise conic arcs themselves. We follow the notation of [10]. The reason we focus on degree 2 NURBS curves is that they are powerful enough to express a wide variety of curves (elliptic, circular, hyperbolic, parabolic arcs) while keeping the algebraic complexity within reasonable bounds at the same time (comparison of algebraic numbers of degree 184 in the worst case). Given three control points $\mathbf{P_0}$, $\mathbf{P_1}$, $\mathbf{P_2}$ and a shape factor s (cf. Fig. 1), we represent a conic arc, parametrized by t, from $\mathbf{P_0}$ to $\mathbf{P_2}$ through $\mathbf{P_1}$ as

$$\mathbf{C}(t) = \frac{(1-t)^2\mathbf{P_0} + 2t(1-t)\frac{s}{1-s}\mathbf{P_1} + t^2\mathbf{P_2}}{(1-t)^2 + 2t(1-t)\frac{s}{1-s} + t^2},\ t \in [0,1].$$

As parameter t traces the arc, we assume that the interior of the shape lies always on the left. Therefore, convex boundary arcs have positive or CCW orientation, while concave boundary curves have negative or CW orientation. The control points have the property that the segments $\mathbf{P_0P_1}$ and $\mathbf{P_1P_2}$ are tangent to $\mathbf{P_0}$ and $\mathbf{P_2}$ respectively. Note that $\mathbf{C}(0) = \mathbf{P_0}$ and $\mathbf{C}(1) = \mathbf{P_2}$. When $s < 1/2$ we have an ellipse, when

*INRIA Nancy Grand-Est

$s = 1/2$ we have a parabola, and $1/2 < s < 1$ yields a hyperbola, as shown in Fig. 1.

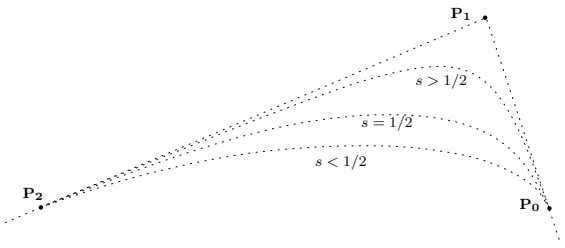


Figure 1: Definition of a quadratic NURBS arc.

3 Bisector curves

Each point on the medial axis is the center of a disk tangent to at least two points of the boundary of the shape. Therefore, the points of the medial axis lie on bisector curves of planar rational curves, or on bisector curves of a point and a curve. They may also lie on the self-bisector of a single curve. The bisector curve of planar rational curves is an algebraic curve and can be defined implicitly in the parametric space as the common intersection of the normal lines at the footpoints and the segment bisector of the footpoints themselves [5]. It is of lower degree than the implicit equation in the cartesian space. We can verify that for conics, the bisector equation in the parametric space is of total degree 12 in the worst case, 6 in each parameter. See [6] for the case of ellipses and [9, 5] for a study of point-curve or curve-curve bisectors. We represent a point on the medial axis by the corresponding footpoints on the boundary curves, therefore we can represent parts of the medial axis by properly trimming the bisector curves. In [13] the bisector curves are split in monotone pieces by using a subdivision technique. In the section that follows we improve this technique by proving and exploiting some algebraic and geometric properties of the bisector curves themselves. This is a generalization of the properties observed and exploited in [6].

4 Maximal bitangent disk

The boundary of the shape consists of two types of arcs. Convex and concave ones. We consider the general case where the tangency points of a maximal bitangent disk are not a concave joint. Let a_1 and a_2 be two arcs and $\mathbf{P_1}$ and $\mathbf{P_2}$ the corresponding footpoints of a maximal bitangent disk B (Fig. 2). Given $\mathbf{P_1}$, since the bisector curve is of degree 6 in each parameter, there may be up to 6 candidate points for $\mathbf{P_2}$. We can select the proper solution by applying the following lemmas.

Lemma 1 *If a_1 is concave, then the tangent line T of a_1 at $\mathbf{P_1}$ separates $\mathbf{P_2}$ and a_1 (Fig. 2 cases (i) and (ii)). If a_1 is convex, then the circle of curvature of a_1 at $\mathbf{P_1}$ contains $\mathbf{P_2}$ (Fig. 2 cases (iii) and (iv)).*

Proof. If a_1 is concave, then a_1 and the maximal disk B have opposite curvatures. Therefore all points of a_1 (except $\mathbf{P_1}$) and B are separated by T. (In this case, T can be seen as a circle with an infinite radius.) If a_1 is convex, then the circle of curvature of a_1 at $\mathbf{P_1}$ is the circle of maximum radius that lies locally inside a_1. Therefore its radius bounds from above the radius of B. □

Applying the above lemma for $\mathbf{P_2}$ yields the following.

Lemma 2 *If a_2 is concave (Fig. 2 cases (i) and (iii)), we consider the tangent lines of a_2 along the arc. At some point $\mathbf{Q}$, the tangent line may pass through $\mathbf{P_1}$ and this will be a boundary condition, that is before $\mathbf{Q}$, $\mathbf{P_1}$ will be on the correct side (left) of the tangent line, while after $\mathbf{Q}$, $\mathbf{P_1}$ will be on the right of the tangent line, thus points after $\mathbf{Q}$ on a_2 are rejected. If a_2 is convex (Fig. 2 cases (ii) and (iv)), we consider the cirles of curvature of a_2 along the arc. At some point $\mathbf{Q}$, the circle of curvature may pass through $\mathbf{P_1}$ and this will be a boundary condition, that is before $\mathbf{Q}$, $\mathbf{P_1}$ will be inside the circle of curvature, while after $\mathbf{Q}$, $\mathbf{P_1}$ will be outside the circle of curvature, therefore points after $\mathbf{Q}$ on a_2 are rejected.*

Algebraically, the tangent line of a_1 evaluated at a point of a_2 is a bivariate polynomial of total degree 4, 2 in each parameter. The circle of curvature of a_1, evaluated at a point of a_2 is a bivariate polynomial of total degree 10, 6 in t_1 and 4 in t_2. Fig. 3 (i) shows an instance of Fig. 2, case (iii), where a_1 is a convex arc and a_2 is a concave one. The graph of Fig. 3 (ii) shows a plot of the parametric bisector (solid line graph). The dashed line graph is a plot of the circle of curvature of a_1 evaluated at a point of a_2. Note that the bisector critical points are separated by the graph of the circle of curvature, providing automatically a subdivision in monotone pieces. This is because at those critical points, the circle of curvature is also a maximal bitangent disk. Similar properties hold for all cases of Fig. 2. Location of the bisector extrema via the circle of curvature is more efficient, because purely algebraic techniques study the bisector curve itself via a discriminant, which has some spurious factors [2]. The reason is that the bisector is not an arbitrary curve but it is computed as some resultant of an algebraic system [5, 6, 7].

5 Critical points

Three-prong points (Fig. 4). A three-prong point on the medial axis is a point where the associated maximal disk is tangent to three points of the boundary of

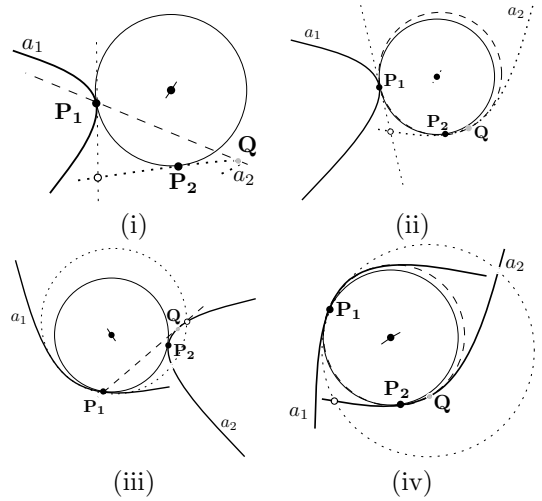


Figure 2: Isolating the proper footpoint.

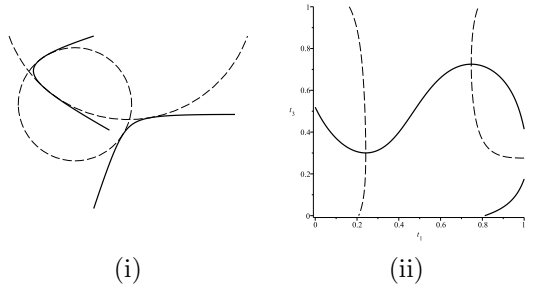


Figure 3: (i) A convex and a concave arc with the circles of curvature corresponding to parametric bisector extrema. (ii) Parametric bisector and circle of curvature evaluated.

the shape. The points of tangency can be described by an algebraic system. The system can be formulated and solved in a way similar to [7].

Terminal points. They are associated with the circle of curvature at points where the curvature attains a local maximum. These can be considered as degenerate cases of (iii), (iv) and (v) in Fig. 4.

Arc joints. These points are in fact artifacts, due to the fact that a single curve is considered as two or more subcurves. Thus, combinatorially, one has to take into account the MA points corresponding to the arc joints.

Let a_1, a_2, a_3 be three conic arcs on the boundary of the shape, with parameters t_1, t_2, t_3 respectively. We express a tritangent circle to a_1, a_2, a_3 by an algebraic system. Footpoint parameter value t_1 can be expressed as a resultant polynomial the degree of which is shown in table 1. The most difficult case is for three conic arcs, where the degree of the polynomial is 184, as in the case for the Voronoi diagram of ellipses [6]. Table 1 also summarizes the simpler cases, when the second footpoint lies on the same arc, or when one footpoint corresponds to an arc endpoint and therefore it is a fixed rational point. The equations used to describe each algebraic system are i) the normal lines N_i at the footpoint of arc a_i, ii) the segment bisector M_{ij} which is perpendicular to the segment joining the footpoints of a_i (or endpoint p_i) and a_j (or p_j), iii) the bisector curve B_{ij} of arcs a_i and a_j or the self-bisector curve B_{ii} of arc a_i.

Computing the footpoint as an algebraic number allows for detection of n-prong points, that is bifurcation points of degree ≥ 3, simply by comparing such algebraic numbers. For example, if there exist a maximal disk tangent to a_1, a_2, a_3, a_4 then the footpoints of the tritangent disks of a_1, a_2, a_3 and a_1, a_2, a_4 on arc a_1 are identical.

Finally, the solution of the algebraic systems of table 1 can be accelerated by incorporating the subdivision technique of [8], because after applying lemmas 1 and 2, the bisectors are split in monotone parts.

	t_2	t_3	deg.	equations				
(i)	a_2	a_3	184	N_1	N_2	N_3	M_{12}	M_{13}
(ii)	p_2	a_3	36	N_1	N_3	M_{12}	M_{23}	
(iii)	a_1	a_2	32	N_1	N_2	M_{12}	B_{11}	
(iv)	a_2	a_2	16	N_1	N_2	M_{12}	M_{22}	B_{22}
(v)	a_1	p_2	8	B_{12}	B_{22}			
(vi)	p_2	p_3	6	N_1	M_{12}	M_{13}		

Table 1: Degree of footpoints for 3-prong MA points (Fig. 4.)

6 Computing the medial axis

The MA is constructed in a way similar to [12]. Instead of tracing the MA via the boundary of the shape with a predefined precision, we check explicitly each critical point, expressed with an algebraic system, as discussed in the previous section. Thus, we trace the MA in a constant number of steps, depending only on the number of input arcs (and not on some precision threshold).

We conclude with an example applying the above techniques. Consider the 9 control points $\mathbf{P_0} \ldots \mathbf{P_7}, \mathbf{P_8} \equiv \mathbf{P_0}$ with coordinates $(1,5)$, $(-5,3)$, $(2,-1)$, $(0,-1)$, $(1,-6)$, $(3,0)$, $(10,0)$, $(3,3)$, $(1,5)$ defining 4 arcs with shape factors $2/3$, $1/4$, $4/5$, $3/5$ respectively, where arc a_{i+1} has control points $(\mathbf{P_{2i}}, \mathbf{P_{2i+1}}, \mathbf{P_{2i+2}})$, $i = 0 \ldots 3$, as shown in Fig. 5. We may start from a convex vertex (i.e., $\mathbf{P_8}$) and trace the boundary of the shape, checking for critical points of the medial axis. This way the medial axis is traced edge by edge. Note that there exist two tritangent disks of a_1,a_3,a_4. However, the dashed one is rejected, as its associated footpoint on a_4 lies after the footpoint of the other tritangent disk, as we move from $\mathbf{P_8}$ to $\mathbf{P_0}$.

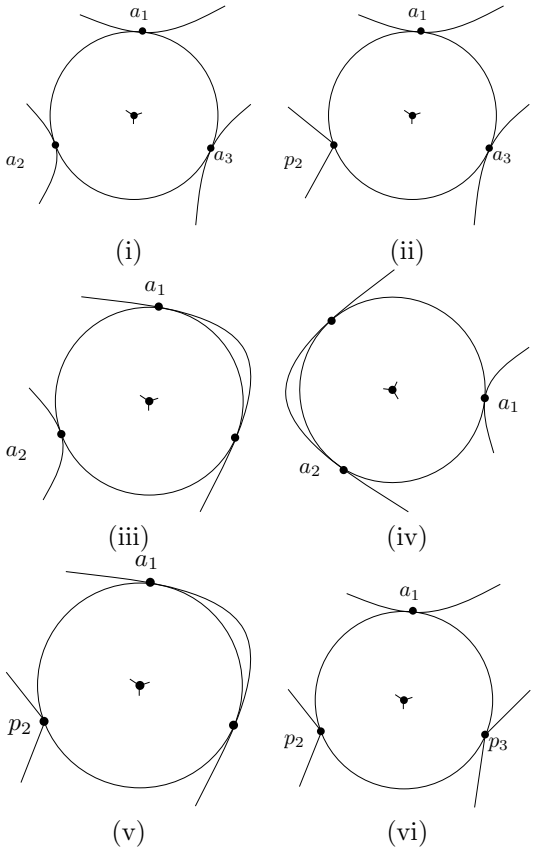


Figure 4: Various types of three-prong points

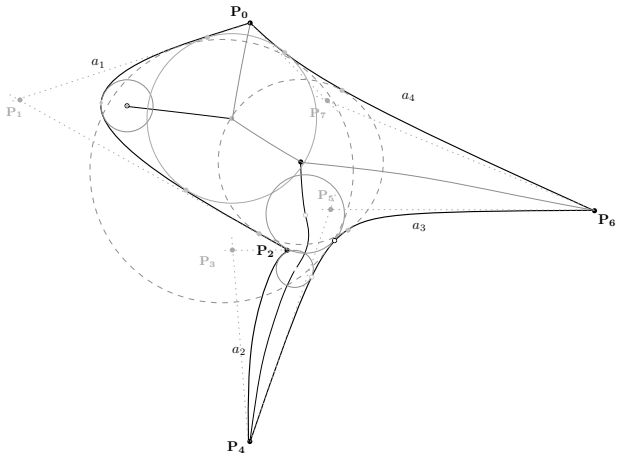


Figure 5: Medial axis of a shape and the maximal disks at its critical points.

References

[1] Harry Blum. A Transformation for Extracting New Descriptors of Shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[2] L. Busé and B. Mourrain. Explicit factors of some iterated resultants and discriminants. *Mathematics of Computations*, 78:345–386, 2009.

[3] Hyeong In Choi, Sung Woo Choi, and Hwan Pyo Moon. Mathematical theory of medial axis transform. *Pacific Journal of Mathematics*, 181(1):57–88, 1997.

[4] J. J. Chou. Voronoi diagrams for planar shapes. *IEEE Comput. Graph. Appl.*, 15(2):52–59, 1995.

[5] G. Elber and Myung-Soo Kim. Bisector curves of planar rational curves. *Comp.-Aid. Des.*, 30:1089–1096, 1998.

[6] I. Z. Emiris, E. P. Tsigaridas, and G. M. Tzoumas. Predicates for the exact Voronoi diagram of ellipses under the euclidean metric. *Intern. J. Comp. Geom. & Appl.*, 18(6):567–597, 2008.

[7] I.Z. Emiris, E.P. Tsigaridas, and G.M. Tzoumas. Exact Delaunay graph of smooth convex pseudocircles: general predicates, and implementation for ellipses. In *SPM '09: 2009 SIAM/ACM Joint Conf. on Geom. & Phys. Model.*, pages 211–222, San Francisco, CA, USA, 2009.

[8] I.Z. Emiris and G.M. Tzoumas. Exact and efficient evaluation of the InCircle predicate for parametric ellipses and smooth convex objects. *Comp.-Aid. Des.*, 40(6):691–700, 2008.

[9] R.T. Farouki and R. Ramamurthy. Degenerate point/curve and curve/curve bisectors arising in medial axis computations for planar domains with curved boundaries. *Computer Aided Geometric Design*, 15:615–635, 1998.

[10] Les Piegl and Wayne Tiller. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[11] R. Ramamurthy and R. T. Farouki. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries - II: Detailed algorithm description. *J. Comput. Appl. Math.*, 102(2):253–277, 1999.

[12] M. Ramanathan and B. Gurumoorthy. Constructing medial axis transform of planar domains with curved boundaries. *Comp.-Aid. Des.*, 35(7):619–632, 2003.

[13] J.-K. Seong, E. Cohen, and G. Elber. Voronoi diagram computations for planar NURBS curves. In *Proc. 2008 ACM Symp. Solid and Phys. Modeling*, pages 67–77, New York, NY, USA, 2008. ACM.

Online Hitting Sets In A Geometric Setting Via Vertex Ranking

Guy Even* Shakhar Smorodinsky†

Abstract

We consider the problem of hitting sets online. The hypergraph (i.e., range-space consisting of points and ranges) is known in advance. However, the ranges to be stabbed are input one-by-one in an online fashion. The online algorithm must stab each range upon arrival. An online algorithm may add points to the hitting set but may not remove already chosen points. The goal is to use the smallest number of points. The best known competitive ratio for online hitting sets by Alon et al. [1] is $O(\log n \cdot \log m)$ for general hypergraphs, where n and m denote the number of points and the number of ranges, respectively. We consider three special classes of hypergraphs.

The first setting consists of subsets of nodes of a given graph that induce connected subgraphs. We show how vertex ranking can be employed to design a simple online algorithm, the competitive ratio of which equals the number of colors used by the vertex ranking. When the underlying graph is a planar graph (e.g., a Delaunay triangulation) with n vertices, we obtain an optimal $O(\sqrt{n})$-competitive ratio. We remark that the analysis of the competitive ratio of the algorithm of [1] only proves an $O(n)$-competitive ratio for this case.

In the second setting, we consider subsets of a given set of n points in the Euclidean plane that are induced by half-planes. We apply the first setting to obtain an $O(\log n)$-competitive ratio. We also prove an $\Omega(\log n)$ lower bound for the competitive ratio in this setting.

In the third setting, we consider subsets of a given set of n points in the plane induced by unit discs. Since the number of subsets in this setting is $O(n^2)$, the competitive ratio obtained by Alon et al. is $O(\log^2 n)$. We introduce an algorithm with $O(\log n)$-competitive ratio. We also show that any online algorithm for this problem has a competitive ratio of $\Omega(\log n)$, and hence our algorithm is optimal.

1 Introduction

In the minimum hitting set problem, we are given a hypergraph (X, R), where X is the ground set of points and R is a set of hyperedges. The goal is to find a finite set $S \subseteq X$ such that every hyperedge is stabbed by S, namely, every hyperedge has a nonempty intersection with S.

The minimum hitting set problem is a classical NP-hard problem [15], and remains hard even for geometrically induced hypergraphs (see [12] for several references). A sharp logarithmic threshold for hardness of approximation was proved by Feige [11]. On the other hand, the greedy algorithm achieves a logarithmic approximation ratio [14, 7]. Better approximation ratios have been obtained for several geometrically induced hypergraphs using specific properties of the induced hypergraphs [12, 17, 2]. Other improved approximation ratios are obtained using the theory of VC-dimension and ε-nets [4, 10, 8]. Much less is known about online versions of the hitting set problem.

In this paper, we consider an online setting in which the set of points X is given in the beginning, and the ranges are introduced one by one. Upon arrival of a new range, the online algorithm may add points (from X) to the hitting set so that the hitting set also stabs the new range. However, the online algorithm may not remove points from the hitting set. We use the competitive ratio, a classical measure for the performance of online algorithms [19, 3], to analyze the performance of online algorithms.

Alon et al. [1] considered the online set-cover problem for arbitrary hypergraphs. In their setting, the ranges are known in advance, and the points are introduced one by one. Upon arrival of an uncovered point, the online algorithm must choose a range that covers the point. Hence, by replacing the roles of ranges and points, their setting is equivalent to our setting. The online set cover algorithm presented by Alon et al. [1] achieves a competitive ratio of $O(\log n \log m)$ where n and m are the number of points and the number of hyperedges respectively. Note that if $m \geq 2^{n/\log n}$, the analysis of the online algorithm only guarantees that the competitive ratio is $O(n)$; a trivial bound if one range is chosen for each point. On the other hand, in many geometric settings the underlying hypergraph has a bounded VC-dimension which implies that the number of hyperedges is polynomial in the number of points (see, e.g., [18]). If m is polynomial in n, their algorithm achieves a competitive ratio of $O(\log^2 n)$. Since there is no matching lower bound of $\Omega(\log^2 n)$ for the competitive ratio, it might be the case that there is an algorithm with an $O(\log n)$-competitive

*School of Electrical Engineering, Tel-Aviv Univ., Tel-Aviv 69978, Israel. guy@eng.tau.ac.il

†Mathematics Department, Ben-Gurion Univ., Be'er Sheva 84105, Israel. shakhar@math.bgu.ac.il

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

ratio for such hypergraphs.

We present online hitting set algorithms for special classes of hypergraphs, some with bounded VC-dimension and some with linear VC-dimension. These algorithms are based on a novel relation between vertex ranking [16] and hitting sets.

The first class of hypergraphs that we consider, is induced by connected components of a given graph. The input of the algorithm is a graph $G = (V, E)$, and the adversary chooses subsets $V' \subseteq V$ such that the induced subgraph $G[V']$ is connected. An application for such a setting is the placement of servers in virtual private networks (VPNs). Each VPN is a subset of vertices that induce a connected subgraph, and requests for VPNs arrive online. The algorithm selects a location for each VPN, and the goal is to select as few servers as possible.

For the case of hypergraphs induced by connected components of graph, we show that one can use vertex ranking to design an efficient online algorithm where the competitive ratio equals the number of colors used by a vertex ranking. In particular, for forests on n vertices, our algorithm achieves an optimal $O(\log n)$-competitive ratio, and for planar graphs our algorithm achieves $O(\sqrt{n})$-competitive ratio. This class is of particular interest since the VC-dimension of such a hypergraph is not bounded. For example, if G is a star (i.e., a vertex v with $n-1$ neighbors), the number of subsets of vertices that induce a connected graph is 2^{n-1}. However, the star has a vertex ranking that uses just two colors, hence, the competitive ratio of our algorithm in this case is 2. This is easily seen to be the best competitive ratio that can be achieved. If G is a planar graph, then G admits a vertex ranking that uses $O(\sqrt{n})$ colors, and the competitive ratio of our algorithm is $O(\sqrt{n})$. This is an improvement over the analysis of the algorithm of Alon et al. [1] which only proves a competitive ratio of $O(n)$. Thus, our algorithm is useful even in hypergraphs whose VC-dimension is unbounded.

Two more classes of hypergraphs are obtained geometrically as follows. In both settings we are given a set X of n points in the plane. In one hypergraph, the hyperedges are intersections of X with half planes. In the other hypergraph, the hyperedges are intersections of X with unit discs. Our main result is an online algorithm for the hitting set problem for points in the plane and unit discs (or half-planes) with an optimal competitive ratio of $O(\log n)$. The competitive ratio of this algorithm improves the $O(\log^2 n)$-competitive ratio of Alon et al. by a logarithmic factor. An application for points and unit discs is the selection of access points or base stations in a wireless network. The points model base stations and the disc centers model clients. The reception range of each client is a disc, and the algorithm has to select a base station that serves a new uncovered client. The goal is to select as few base stations as possible.

2 Preliminaries

Let (X, R) denote a hypergraph, where R is a set of nonempty subsets of the ground set X. Members in X are referred to as *points*, and members in R are referred to as *ranges* (or *hyperedges*). A subset $S \subseteq X$ *stabs* a range r if $S \cap r \neq \emptyset$. A *hitting set* is a subset $S \subseteq X$ that stabs every range in R. In the minimum hitting set problem, the goal is to find a hitting set with the smallest cardinality.

In this paper, we consider the following online setting. The adversary introduces a sequence $\sigma \triangleq \{r_i\}_{i=1}^{s}$ of ranges. Let σ_i denote the prefix $\{r_1, \ldots, r_i\}$. The online algorithm computes a chain of hitting sets $C_1 \subseteq C_2 \subseteq \cdots$ such that C_i is a hitting set with respect to the ranges in σ_i.

The competitive ratio of the algorithm is defined as follows. Let $\textsc{opt}(\sigma) \subseteq X$ denote a minimum cardinality hitting set for the ranges in σ. Let $\textsc{alg}(\sigma) \subseteq X$ denote the hitting set computed by an online algorithm ALG when the input sequence is σ. Note that the sequence of minimum hitting sets $\{\textsc{opt}(\sigma_i)\}_i$ is not necessarily a chain of inclusions. The *competitive ratio* of an online hitting set algorithm ALG is defined as the supremum, over all sequences σ of ranges, of the ratio $|\textsc{alg}(\sigma)|/|\textsc{opt}(\sigma)|$.

3 Summary of Our Results

Connected Subgraphs. We consider the following setting of a hypergraph induced by connected subgraphs of a given graph. Formally, let $G = (V, E)$ be a graph. Let $H = (V, R)$ denote the hypergraph over the same set of vertices V. A subset $r \subseteq V$ is a hyperedge in R if and only if the subgraph $G[r]$ induced by r is connected.

We need the notion of a vertex ranking of a graph[16]. A *vertex ranking* is a function $c : V \to \mathbb{N}$ that satisfies the following property: For any pair of vertices $u, v \in V$, if $c(u) = c(v)$, then, for any simple path P in G connecting u and v, there is a vertex w in P such that $c(w) > c(u)$. Notice that, in particular, a vertex ranking of a graph G is also a proper coloring of G since adjacent vertices must get distinct colors. For a subset $X' \subset X$, let $c_{\max}(X') \triangleq \max\{c(v) \mid v \in X'\}$. It is easy to see that if c is a vertex ranking, then $|\{v \in r \mid c(v) = c_{\max}(r)\}| = 1$, for every $r \in R$. Thus, let $v_{\max}(r)$ denote the (unique) vertex v in r such that $c(v) = c_{\max}(r)$.
Our first result is an online hitting set algorithm for connected subgraphs.

Theorem 1 *Let $c : V \to \mathbb{N}$ denote a vertex ranking of a graph $G = (V, E)$. Then there exists an online*

hitting set algorithm for the connected subgraphs of G with a competitive ratio of $c_{\max}(V)$.

By [16], planar graphs admit vertex rankings with $c_{\max}(V) = O(\sqrt{|V|})$. Therefore, Theorem 1 implies that the competitive ratio of our algorithm for connected subgraphs of planar graphs is $O(\sqrt{n})$. We also prove that this competitive is optimal.

Theorem 2 *The competitive ratio of every online hitting set algorithm for connected subgraphs of planar graphs is $\Omega(\sqrt{n})$.*

Points and Half-Planes. We prove the following results for hypergraphs in which the ground set X is a finite set of n points in $\mathbb{R}^2$ and the ranges are all subsets of X that can be cut off by a half-plane. Namely, each range r is induced by a line L_r such that r is the set of points of X in the half-plane below (respectively, above) the line L_r.

Theorem 3 *The competitive ratio of every online hitting set algorithm for points and half-planes is $\Omega(\log n)$.*

Theorem 4 *There exists an online hitting set algorithm for points and half-planes that achieves a competitive ratio of $O(\log n)$.*

Points and Congruent Discs. We prove the following results for hypergraphs in which the ground set X is a finite subset of n points in $\mathbb{R}^2$ and the ranges are intersections of X with unit discs. Namely, a unit disc d induces a range $r = r(d)$ defined by $r = d \cap X$.

Theorem 5 *The competitive ratio of every online hitting set algorithm for points and unit discs is $\Omega(\log n)$.*

Theorem 6 *There exists an online hitting set algorithm for points and unit discs that achieves a competitive ratio of $O(\log n)$.*

4 Vertices and Connected Subgraphs

4.1 Algorithm Description

A listing of Algorithm HS appears as Algorithm 1. The algorithm is input a graph $G = (V, E)$ and a vertex ranking $c : V \to \mathbb{N}$. The sequence $\sigma = \{r_i\}_i$ of subsets of vertices that induce connected subgraphs is input online.

Algorithm 1 HS(G, c) - an online hitting set for connected subgraphs, given a vertex ranking c.

Require: $G = (V, E)$ is a graph and $c : V \to \mathbb{N}$ is a vertex ranking.

```
C_0 ← ∅
for i = 1 to ∞ do {arrival of a range r_i}
    if r_i is not stabbed by C_{i-1} then
        C_i ← C_{i-1} ∪ {v_max(r_i)} {add the vertex with the max color in r_i}
    else
        C_i ← C_{i-1}
    end if
end for
```

4.2 Analysis of The Competitive Ratio

Definition 1 *For a color a, let $\sigma(a)$ denote the subsequence of σ that consists of ranges that satisfy: (i) r_i is not stabbed by C_{i-1}, and (ii) $c_{\max}(r_i) = a$.*

The following lemma implies a lower bound on the (offline) minimum hitting set of the ranges in $\sigma(a)$.

Lemma 7 *If $r_i, r_j \in \sigma(a)$, then the subgraph $G[r_i \cup r_j]$ induced by $r_i \cup r_j$ is not connected. Hence, the ranges in $\sigma(a)$ are pairwise disjoint.*

Proof. Clearly, $c_{\max}(r_i \cup r_j) = \max\{c_{\max}(r_i), c_{\max}(r_j)\} = a$. Assume that $r_i \cup r_j$ induces a connected subgraph. Since c is a vertex ranking, we conclude that $r_i \cup r_j$ contains exactly one vertex colored a. This implies that $v_{\max}(r_i) = v_{\max}(r_j)$. If $j > i$, then the range r_j is stabbed by C_{j-1} since it is stabbed by C_i, a contradiction. □

Proof. [Proof of Theorem 1] Algorithm HS satisfies $|\text{HS}(\sigma)| = \sum_{a\in\mathbb{N}} |\sigma(a)|$. But $\sum_{a\in\mathbb{N}} |\sigma(a)| \le c_{\max}(V) \cdot \max_{a\in\mathbb{N}} |\sigma(a)|$. By Lemma 7, each range in $\sigma(a)$ must be stabbed by a distinct vertex, thus $|\text{OPT}(\sigma)| \ge \max_{a\in\mathbb{N}} |\sigma(a)|$, and the theorem follows. □

Corollary 8 *Let $G = (V, E)$ be a planar graph and let H be the hypergraph consisting of V together with all subsets of vertices inducing connected subgraphs. Then there is an online hitting set for H with competitive ratio of $O(\sqrt{n})$.*

Proof. The proof uses the fact that there exists a vertex ranking for G with a total of $O(\sqrt{n})$ colors [16]. Combining such a vertex ranking with Algorithm HS and using Theorem 1 completes the proof. □

Remark 1 *The above corollary applies to arbitrary graphs with small balanced separators. Let $G = (V, E)$ be a graph such that every subgraph with m vertices has a balanced separator with $O(m^\alpha)$ vertices, for some fixed $0 < \alpha \le 1$. Then there is an*

online hitting set algorithm for the connected subgraphs with competitive ratio of $O(n^\alpha)$. The proof uses all of the above mentioned ingredients, replacing $\sqrt{n}$ with n^α.

For the special case of trees, since there is always a balanced separator with 1 vertex, it is easily seen that trees admit vertex ranking with $O(\log n)$ colors and hence, Algorithm HS *achieves competitive ratio of $O(\log n)$. This bound is optimal as it holds as a lower bound even when the tree is a simple path.*

5 Open Problems

The main challenge left in this paper is to extend the result for unit discs to arbitrary discs. We conjecture that there exists an online hitting set algorithm for hitting arbitrary discs with a subset of a given set of n points in the plane, which has competitive ratio of $O(\log n)$. A more difficult problem is to design an online hitting set algorithm with a logarithmic competitive ratio for any hypergraph with bounded VC-dimension. To the best of our knowledge, there is no better lower bound except for $\Omega(\log n)$ for this problem.

References

[1] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The Online Set Cover Problem. *SIAM Journal on Computing*, 39:361, 2009.

[2] B. Ben-Moshe, M. Katz, and J. Mitchell. A constant-factor approximation algorithm for optimal terrain guarding. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 515–524. Society for Industrial and Applied Mathematics, 2005.

[3] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*, volume 2. Cambridge University Press Cambridge, 1998.

[4] H. Brönnimann and M. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry*, 14(1):463–479, 1995.

[5] P. Cheilaris and S. Smorodinsky. Choosability in geometric hypergraphs. *Arxiv preprint arXiv:1005.5520*, 2010.

[6] K. Chen, H. Kaplan, and M. Sharir. Online conflict-free coloring for halfplanes, congruent disks, and axis-parallel rectangles. *ACM Transactions on Algorithms (TALG)*, 5(2):1–24, 2009.

[7] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[8] K. Clarkson and K. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete and Computational Geometry*, 37(1):43–58, 2007.

[9] G. Even, Z. Lotker, D. Ron, and S. Smorodinsky. Conflict-Free Colorings of Simple Geometric Regions with Applications to Frequency Assignment in Cellular Networks. *SIAM Journal on Computing*, 33:94, 2003.

[10] G. Even, D. Rawitz, and S. Shahar. Hitting sets when the VC-dimension is small. *Information Processing Letters*, 95(2):358–362, 2005.

[11] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[12] D. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM (JACM)*, 32(1):136, 1985.

[13] A. Iyer, H. Ratliff, and G. Vijayan. Optimal node ranking of trees. *Information Processing Letters*, 28(5):225–229, 1988.

[14] D. Johnson. Approximation algorithms for combinatorial problems*. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[15] R. Karp. Reducibility among combinatorial problems, 85–103. In *Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, NY*, 1972.

[16] M. Katchalski, W. McCuaig, and S. Seager. Ordered colourings. *Discrete Mathematics*, 142(1-3):141–154, 1995.

[17] V. Kumar and H. Ramesh. Covering rectilinear polygons with axis-parallel rectangles. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 445–454. ACM, 1999.

[18] J. Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[19] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

Connecting a Set of Circles with Minimum Sum of Radii

Erin Wolf Chambers* Sándor P. Fekete † Hella-Franziska Hoffmann† Dimitri Marinakis‡§
Venkatesh Srinivasan§ Ulrike Stege§ Sue Whitesides§

Abstract

We consider the problem of assigning radii to given points in the plane, such that the resulting set of circles is connected, and the sum of radii is minimized. We show that the problem is polynomially solvable if a connectivity tree is given, but NP-hard if there are upper bounds on the radii; the case of unbounded radii is an open problem. We also give approximation guarantees for a variety of heuristics, describe upper and lower bounds (which are matching in some of the cases), and conclude with experimental results.

1 Introduction

We consider a natural geometric connectivity problem, arising from the context of assigning ranges to a set of center points. More formally, given a set of points $P = \{p_1, \ldots, p_n\}$ in the plane and their respective radii r_i, in the connectivity graph $G = \{V, E\}$, V corresponds to P, and edges $e_{ij} \in E$ to intersecting circles, i.e., $r_i + r_j \geq \text{dist}(p_i, p_j)$ for the Euclidean distance $\text{dist}(p_i, p_j)$ between p_i and p_j. (A natural generalization arises from considering distances in a given weighted graph, instead of geometric distances.) The Connected Range Assignment Problem (CRA) requires assignment of radii r to P, such that the objective function $Q = \sum_{i=1}^{n} r_i^{\alpha}, \alpha = 1$ is minimized, subject to the constraint that G is connected.

Problems of this type have been considered before and have natural motivations from fields including networks, robotics, and data analysis. Common to most is an objective function that considers the sum of the radii of circles to some exponent α.

Alt *et al.* [1] consider the closely related problem of selecting circle centers and radii such that a given set of points in the plane are covered by the circles. Like our work, they focus on minimizing an objective function based on $\sum_i r_i^{\alpha}$ and produce results specific to various values of α. The minimum sum of radii circle coverage problem (with $\alpha = 1$) is also considered by Lev-Tov and Peleg [6] in the context of radio networks. Related work has also been done in the area of data clustering. Gibson *et al.* [5], consider partitioning data into k clusters to minimize the sum of the cluster radii, and authors consider the problem for specific numbers of dimensions. Since we are given the circle centers, the problem can be also considered a *range assignment problem* [3]; see [4] for hardness results of different (typically directed) *communication graphs*.

In this paper we present a variety of algorithmic aspects of the problem. In Section 2 we show that for a given connectivity tree, an optimal solution can be computed efficiently. Section 3 sketches a proof of NP-hardness for the problem when there is an upper bound on the radii. Section 4 provides a number of approximation results for the case of unbounded radii, complemented by experiments in Section 5.

2 CRA for a Given Connectivity Tree

For a given connectivity tree, our problem is polynomially solvable, based on the following observation.

Lemma 1 *Given a connectivity tree T with at least three nodes. There exists an optimal range assignment for T with $r_i = 0$ for all leaves p_i of T.*

Proof. Assume an optimal range assignment for T has a leaf $p_i \in P$ with radius $r_i > 0$. The circle C_i around p_i intersects circle C_j around p_i's parent p_j with radius r_j. Extending C_j to $r_j := \text{dist}(p_i, p_j)$ while setting $r_i := 0$ does not increase $\sum_{p_i \in P} r_i$. □

Direct consequences of Lemma 1 are the following.

Corollary 2 *There is an optimal range assignment satisfying Lemma 1 and further $r_j > 0$ for all $p_j \in P$ of height 1 in T (i.e., each p_j is parent of leaves only).*

Corollary 3 *Consider an optimal range assignment for T satisfying Lemma 1. Further let $p_j \in P$ be of height 1 in T. Then $r_j \geq \max_{p_i \text{ is child of } p_j} \{dist(p_i, p_j)\}$.*

These observations allow us to solve the problem via dynamic programming; details are omitted.

*Department of Computer Science, Saint Louis University, USA. echambe5@slu.edu.

†Algorithms Group, TU Braunschweig, Braunschweig, Germany. {s.fekete,h-f.hoffmann}@tu-bs.de.

‡Kinsol Research Inc., Duncan, BC, Canada. dmarinak@kinsolresearch.com.

§Department of Computer Science, University of Victoria, Victoria, BC, Canada. {sue,stege,venkat}@uvic.ca.

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

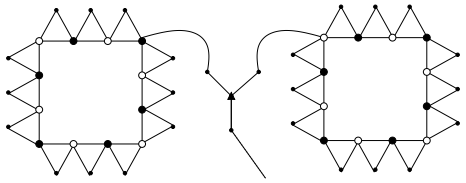


Figure 1: Two variable gadgets connected to the same clause gadget. "True" and "False" vertices marked in bold white or black; auxiliary vertices are indicated by small dots; the clause vertex is indicated by a triangle. Connectivity edges are not shown.

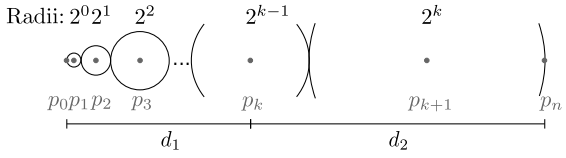


Figure 2: A class of CRA instances that need $k+1$ circles in an optimal solution.

Theorem 4 *For a given connectivity tree, CRA can be solved in polynomial time.*

3 Range Assignment for Bounded Radii

Without a connectivity tree, and assuming an upper bound of ρ on the radii, the problem becomes NP-hard; in this short abstract, we focus on the graph version of the problem.

Theorem 5 *With radii bounded by some constant ρ, the problem CRA is NP-hard in weighted graphs.*

See Figure 1 for the basic construction. The proof uses a reduction from 3Sat. Variable are represented by closed "loops" at distance ρ that have two feasible connected solutions: auxiliary points ensure that either the odd or the even points in a loop get radius ρ. Additional "connectivity" edges ensure that all variable gadgets are connected. Each clause is represented by a star-shaped set of four points that is covered by one circle of radius ρ from the center point. This circle is connected to the rest of the circles, if and only if one of the variable loop circles intersects it, which is the case if and only if there is a satisfying variable.

4 Solutions with a Bounded Number of Circles

In this section we show that using only a small number of circles already yields good approximations; we start by a class of lower bounds.

Theorem 6 *Even for a set of collinear points, a best k-circle solution may be off by a factor of $(1+\frac{1}{2^{k+1}})$.*

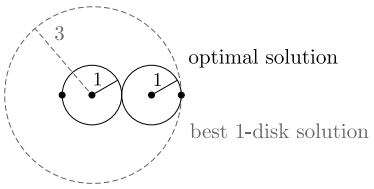


Figure 3: A lower bound of $\frac{3}{2}$ for 1-circle solutions.

Proof. Consider the example in Fig. 2. The provided solution is optimal, as $\sum r_i = \frac{\text{dist}(p_0,p_n)}{2}$. Further, for any integer $k \geq 2$ we have $d_1 = 2\sum_{i=0}^{k-2} 2^i + 2^{k-1} < 2 \cdot 2^k + 2^{k-1} = d_2$. So the radius r_{k+1} cannot be changed in an optimal solution. Inductively, we conclude that exactly $k+1$ circles are needed. Because we only consider integer distances, a best k-circle solution has cost $R_k \geq R+1$, i.e., $\frac{R_k}{R} \geq 1 + \frac{1}{2^{k+1}}$. □

In the following we give some good approximation guarantees for CRA using one or two circles.

Lemma 7 *Let $\mathcal{P}$ a longest path in an optimal connectivity graph, and let e_m be an edge in $\mathcal{P}$ containing the midpoint of $\mathcal{P}$. Then $\sum r_i \geq \max\{\frac{1}{2}|\mathcal{P}|, |e_m|\}$.*

Theorem 8 *A best 1-circle solution for CRA is a $\frac{3}{2}$-approximation.*

Proof. Consider a longest path $\mathcal{P} = (p_0, \ldots, p_k)$ of length $|\mathcal{P}|$ in an optimal connectivity graph. Let $R = \sum r_i$ be the cost of the optimal solution, and $e_m = p_i p_{i+1}$ as in Lemma 7. Let $\bar{d}_i := \text{dist}(p_i, p_k)$ and $\bar{d}_{i+1} := \text{dist}(p_0, p_{i+1})$. Then $\min\{\bar{d}_i, \bar{d}_{i+1}\} \leq \frac{\bar{d}_i + \bar{d}_{i+1}}{2} = \frac{\text{dist}(p_0,p_i)+2|e_m|+\text{dist}(p_{i+1},k)}{2} = \frac{|\mathcal{P}|}{2} + \frac{|e_m|}{2} \leq R + \frac{R}{2} = \frac{3}{2}R$. So one circle with radius $\frac{3}{2}R$ around the point in P that is nearest to the middle of path $\mathcal{P}$ covers P, as otherwise there would be a longer path. □

Fig. 3 shows that this bound is tight. Using two circles yields an even better approximation factor.

Theorem 9 *A best 2-circle solution for CRA is a $\frac{4}{3}$-approximation.*

Proof. Let $\mathcal{P} = (p_0, \ldots, p_k)$ be a longest path in an optimal connectivity graph. Then $\sum r_i \geq \frac{1}{2}|\mathcal{P}|$. We distinguish two cases; see Fig. 4.

Case 1. There is a point x on $\mathcal{P}$ at a distance of at least $\frac{1}{3}|\mathcal{P}|$ from both endpoints. Then there is a 1-circle solution that is a $\frac{4}{3}$-approximation, and no 2-circle solution of such quality is needed.

Case 2. There is no such point x. Let $e_m = p_i p_{i+1}$ be defined as in Lemma 7. Further, let $d_i := \text{dist}(p_0, p_i)$ and $d_{i+1} := \text{dist}(p_{i+1}, p_k)$. Then $|e_m| = |\mathcal{P}| - d_i - d_{i+1}$ and $d_i, d_{i+1} < \frac{1}{3}|\mathcal{P}|$.

Case 2a. If $|e_m| < \frac{1}{2}|\mathcal{P}|$ then $d_i + d_{i+1} = |\mathcal{P}| - |e_m| > \frac{1}{2}|\mathcal{P}| > |e_m|$. Set $r_i := d_i$ and $r_{i+1} = d_{i+1}$,

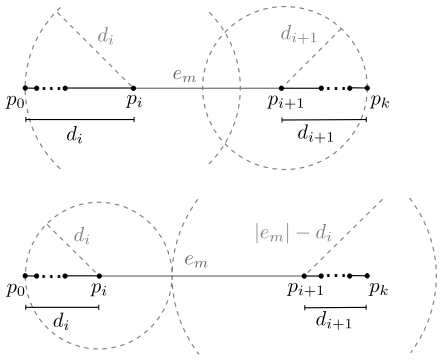


Figure 4: A best 2-circle solution is a $\frac{4}{3}$-approximate solution: Case 2a (Top); Case 2b (Bottom).

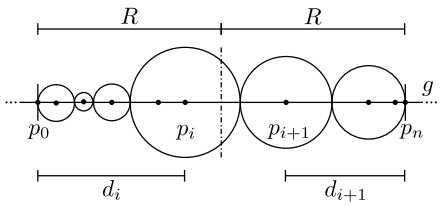


Figure 5: A non-overlapping optimal solution.

then the path is covered. Since $d_i, d_{i+1} < \frac{1}{3}|\mathcal{P}|$ we have $r_i + r_{i+1} = d_i + d_{i+1} < \frac{2}{3}|\mathcal{P}| \leq \frac{4}{3}\sum r_i$ and the claim holds.

Case 2b. Otherwise, if $|e_m| \geq \frac{1}{2}|\mathcal{P}|$ then $d_i + d_{i+1} \leq \frac{1}{2}|\mathcal{P}| \leq |e_m|$. Assume $d_i \geq d_{i+1}$. Choose $r_i := d_i$ and $r_{i+1} := |e_m| - d_i$. Then $\mathcal{P}$ is covered and $r_i + r_{i+1} = d_i + (|e_m| - d_i) = |e_m|$, which is the lower bound and thus the range assignment is optimal. □

If all points of P lie on a straight line, the approximation ratio for two circles can further be improved.

Lemma 10 *Let P be a subset of a straight line. Then there is a non-overlapping optimal solution, i.e., one in which all circles have disjoint interior.*

Proof. An arbitrary optimal solution is modified as follows. For every two overlapping circles C_i and C_{i+1} with centers p_i and p_{i+1}, we decrease r_{i+1}, such that $r_i + r_{i+1} = \text{dist}(p_i, p_{i+1})$, and increase the radius of C_{i+2} by the same amount. This can be iterated, until there is at most one overlap at the outermost circle C_j (with C_{j-1}). Then there must be a point p_{j+1} on the boundary of C_j: otherwise we could shrink C_j contradicting optimality. Decreasing C_j's radius r_j by the overlap l and adding a new circle with radius l around p_{j+1} creates an optimal solution without overlap. □

Theorem 11 *Let P a subset of a straight line g. Then a best 2-circle solution for CRA is a $\frac{5}{4}$-approximation.*

Proof. According to Lemma 10 we are, w.l.o.g., given an optimal solution with non-overlapping circles. Let

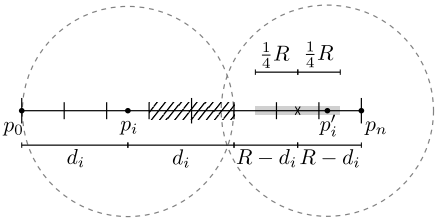


Figure 6: A $\frac{5}{4}$-approximate solution with $d_i < \frac{3}{4}R$. The cross marks the position of the optimal counterpart p_i^* to p_i and the grey area sketches A_i.

p_0 and p_n be the outermost intersection points of the optimal solution circles and g. W.l.o.g., we may further assume $p_0, p_n \in P, R = \sum r_i = \frac{\text{dist}(p_0,p_n)}{2}$ (otherwise, we can add the outermost intersection point of the outermost circle and g to P, which may only improve the approximation ratio). Let p_i denote the rightmost point in P left to the middle of $\overline{p_0p_n}$ and let p_{i+1} its neighbor on the other half. Further, let $d_i := \text{dist}(p_0, p_i)$, $d_{i+1} := \text{dist}(p_{i+1}, p_n)$ (See Fig. 5). Assume, $d_i \geq d_{i+1}$. We now give $\frac{5}{4}$-approximate solutions using one or two circles that cover $\overline{p_0p_n}$.

Case 1. If $\frac{3}{4}R \leq d_i$ then $\frac{5}{4}R \geq 2R - d_i = \text{dist}(p_i, p_n)$. Thus, a one-circle solution around p_i is sufficient.

Case 2. If $\frac{3}{4}R > d_i \geq d_{i+1}$ we need two circles to cover $\overline{p_0p_n}$ with $\frac{5}{4}R$.

Case 2a. The point p_i could be a center point of an optimal two-circle solution if there was a point p_i^* with $\text{dist}(C_i, p_i^*) = \text{dist}(p_i^*, p_n) = R - d_i$. So in case there is a $p'_i \in P$ that lies in a $\frac{1}{4}R$-neighborhood of such an optimal p_i^* we get $\text{dist}(C_i, p'_i), \text{dist}(p'_i, p_n) \leq R - d_i + \frac{1}{4}R$ (see Fig. 6). Thus, $r(p_i) := d_i, r(p'_i) := R - d_i + \frac{1}{4}R$ provides a $\frac{5}{4}$-approximate solution.

Case 2b. Analogously to Case 2a, there is a point $p'_{i+1} \in P$ within a $\frac{1}{4}R$-range of an optimal counterpart to p_{i+1}. Then we can take $r(p_{i+1}) := d_{i+1}$, $r(p'_{i+1}) := R - d_{i+1} + \frac{1}{4}R$ as a $\frac{5}{4}$-approximate solution.

Case 2c. Assume that there is neither such a p'_i nor such a p'_{i+1}. Because d_i, d_{i+1} are in $(\frac{1}{4}R, \frac{3}{4}R)$, we have $\frac{1}{4}R < R - d_j < \frac{3}{4}R$ for $j = i, i+1$, which implies that there are two disjoint areas A_i, A_{i+1}, each with diameter equal to $\frac{1}{2}R$ and excluding all points of P. Because p_i, the rightmost point on the left half of $\overline{p_0p_n}$, has a greater distance to A_i than to p_0, any circle around a point on the left could only cover parts of both A_i and A_{i+1} if it has a greater radius than its distance to p_0. This contradicts the assumption that p_0 is a leftmost point of a circle in an optimal solution. The same applies to the right-hand side. Thus, $A_i \cup A_{i+1}$ must contain at least one point of P, and therefore one of the previous cases leads to a $\frac{5}{4}$-approximation. □

Fig. 7 shows that the bound is tight. We believe that this is also the worst case when points are *not* on

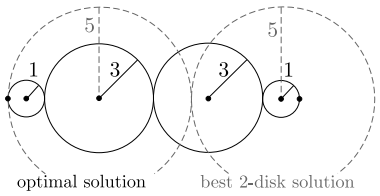


Figure 7: A lower bound of $\frac{5}{4}$ for 2-circle solutions.

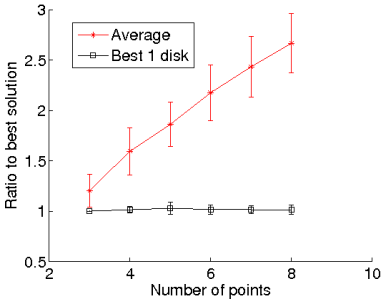


Figure 8: Ratios of the average over all enumerated trees and of the best 1-circle tree to the optimal $\sum r_i$. Results were averaged over 100 trials for each number.

a line. Indeed, the solutions constructed in the proof of Theorem 11 cover a longest path $\mathcal{P}$ in an optimal solution for a general P. If this longest path consists of at most three edges, $p_i(=: p'_{i+1})$ and $p_{i+1}(=: p'_i)$ can be chosen as circle centers, covering all of P. However, if $\mathcal{P}$ consists of at least four edges, a solution for the diameter may produce two internal non-adjacent center points that do not necessarily cover all of P.

5 Experimental Results

It is curious that even in the worst case, a one-circle solution is close to being optimal. This is supported by experimental evidence. For different numbers of uniformly distributed points, we enumerated all possible spanning trees using the method described in [2], and recorded the optimal value with the algorithm mentioned in Section 2. This we compared with the best one-circle solution; as shown in Fig. 8, the latter seems to be an excellent heuristic choice.

6 Conclusion

A number of open problems remain. One of the most puzzling is the issue of complexity in the absence of upper bounds on the radii. The strong performance of the one-circle solution (and even better of solutions with higher, but limited numbers of circles), and the difficulty of constructing solutions for which the one-circle solution is not optimal strongly hint at the possibility of the problem being polynomially solvable. One possible way may be to use methods from linear programming: modeling the objective function and the variables by linear methods is straightforward; describing the connectivity of a spanning tree by linear cut constraints is also well know. However, even though separating over the exponentially many cut constraints is polynomially solvable (and hence optimizing over the resulting polytope), the overall polytope is not necessarily integral. On the other hand, we have been unable to prove NP-hardness without upper bounds on the radii, even in the more controlled context of graph-induced distances.

Other open problems are concerned with the worst-case performance of heuristics using a bounded number of circles. We showed that two circles suffice for a $\frac{4}{3}$-approximation in general, and a $\frac{5}{4}$-approximation on a line; we conjecture that the general performance guarantee can be improved to $\frac{5}{4}$, matching the existing lower bound. Obviously, the same can be studied for k circles, for any fixed k; at this point, the best lower bounds we have are $\frac{7}{6}$ for $k = 3$ and $1 + \frac{1}{2^{k+1}}$ for general k. We also conjecture that the worst-case ratio $f(k)$ of a best k-circle solution approximates the optimal value arbitrarily well for large k, i.e., $\lim_{k\to\infty} f(k) = 1$.

Acknowledgments

This work was started during the 2009 Bellairs Workshop on Computational Geometry. We thank all other participants for contributing to the great atmosphere.

References

[1] H. Alt, E. M. Arkin, H. Brönnimann, J. Erickson, S. P. Fekete, C. Knauer, J. Lenchner, J. S. B. Mitchell, and K. Whittlesey. Minimum-cost coverage of point sets by disks. In *Proc. 22nd Annu. ACM Sympos. Comput. Geom. (SoCG)*, pages 449–458, 2006.

[2] D. Avis and K. Fukuda. Reverse search for enumeration. *Disc. Appl. Math.*, 65(1-3):21–46, 1996.

[3] A. E. Clementi, P. Penna, and R. Silvestri. On the power assignment problem in radio networks. *Mobile Networks and Applications*, 9(2):125–140, 2004.

[4] B. Fuchs. On the hardness of range assignment problems. *Networks*, 52(4):183–195, 2008.

[5] M. Gibson, G. Kanade, E. Krohn, I. A. Pirwani, and K. Varadarajan. On clustering to minimize the sum of radii. In *Proc. 19th ACM-SIAM Symp. Disc. Alg. (SODA)*, pages 819–825, 2008.

[6] N. Lev-Tov and D. Peleg. Polynomial time approximation schemes for base station coverage with minimum total radii. *Computer Networks*, 47(4):489–501, 2005.

A 3-Approximation Algorithm for Computing Partitions with Minimum Stabbing number of Rectilinear Simple Polygons

Mohammad Ali Abam* Boris Aronov† Mark de Berg‡ Amirali Khosravi‡

Abstract

Let P be a rectilinear simple polygon. The stabbing number of a partition of P into rectangles is the maximum number of rectangles stabbed by any axis-parallel segment inside P. We present a 3-approximation algorithm for finding a partition with minimum stabbing number. It is based on an algorithm that finds an optimal partition for histograms.

1 Introduction

Computing decompositions of simple polygons is one of the fundamental problems in computational geometry. When the polygon at hand is arbitrary then one typically wants a decomposition into triangles, and when the polygon is rectilinear one wants a decomposition into rectangles. Sometimes any such decomposition will do; then one can compute an arbitrary triangulation or, for rectilinear polygons, a vertical decomposition. In other cases one would like the decomposition to have certain properties. The property we are interested in is the so-called stabbing number—see below for a definition—of the decomposition.

Let P be a rectilinear simple polygon with n vertices. We call a decomposition of P into rectangles a *rectangular partition*. The stabbing number of a segment s with respect to a rectangular partition R is the number of rectangles intersected by s, and the *(rectilinear) stabbing number* of R is the maximum stabbing number of any axis-parallel segment s in the interior of P. De Berg and Van Kreveld [2] showed that any rectilinear polygon admits a rectangular partition with stabbing number $O(\log n)$. This bound is asymptotically tight in the worst case: any rectangular partition of a staircase polygon with n vertices has stabbing number $\Omega(\log n)$.

*Department of Computer Science, TU Dortmund, 44221 Dortmund, Germany, mohammad-ali.abam@tu-dortmund.de

†Department of Computer Science and Engineering, Polytechnic Institute of NYU, Brooklyn, NY 11201-3840, USA, aronov@poly.edu. Work by Boris Aronov has been supported by grant No. 2006/194 from the U.S.-Israel Binational Science Foundation, by NSF Grant CCF-08-30691, and by NSA MSP Grant H98230-10-1-0210.

‡Department of Computing Science, TU Eindhoven, PO Box 513, 5600 MB Eindhoven, the Netherlands, {mdberg,akhosrav}@win.tue.nl. Work by Amirali Khosravi has been supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 612.000.631.

The algorithm of De Berg and Van Kreveld [2] guarantees partitions which are tight in the worst case. However, some rectilinear polygons admit partitions with stabbing number $O(1)$. This leads to the topic of our paper: finding an *optimal rectangular* partition of a rectilinear polygon P whose stabbing number is minimum over all such partitions.

The problem is not known to be NP-complete. We present a 3-approximation algorithm for it. It is based on an algorithm that finds optimal rectangular partitions for histograms. Due to lack of space all the proofs are omitted in this version, and can be found in the full version of the paper.

Related work. Chazelle *et al.* [3] studied the stabbing number of convex decompositions of polytopes. Tóth showed that any partition of d-dimensional space ($d \geqslant 2$) into n axis-aligned boxes has rectilinear stabbing number $\Omega(\log^{1/(d-1)} n)$, and he presented partitioning scheme achieving this bound [6]. Hershberger and Suri [4] gave an algorithm for triangulating a simple polygon with stabbing number $O(\log n)$ which is worst-case tight. Considering triangulations of point sets, Agarwal, Aronov and Suri [1] proved that one can triangulate n points in $\mathbb{R}^2$ and $\mathbb{R}^3$ (using Steiner points) with stabbing number $O(\sqrt{n} \cdot \log n)$.

2 Optimal rectangular partitions

Let P be a rectilinear simple polygon with n vertices. We denote the interior of P by $\mathrm{int}(P)$ and its boundary by ∂P. In the remainder of paper, whenever we speak of partitions and stabbing numbers, we mean rectangular partitions and rectilinear stabbing numbers. We denote the stabbing number of a partition R by $\sigma(R)$. The *horizontal stabbing number* of R, denoted $\sigma_{\mathrm{hor}}(R)$, is defined as the maximum stabbing number of any horizontal segment $s \subset P$. The *vertical stabbing number*, denoted $\sigma_{\mathrm{vert}}(R)$, is defined similarly. Note that $\sigma(R) = \max(\sigma_{\mathrm{hor}}(R), \sigma_{\mathrm{vert}}(R))$.

We start by studying the properties of optimal partitions. Consider a partition R of P. The partition is induced by a set $E(R)$ of *maximal edges*, that is, segments of maximal length that are the union of one or more rectangle edges that are not part of ∂P. A maximal edge is *anchored* if at least one of its endpoints is a vertex of P. We first show that there exists an optimal partition with only anchored edges.

Lemma 1 *Any rectilinear simple polygon P has an optimal partition R_{opt} in which all maximal edges are anchored.*

A *binary space partition*, or BSP for short, of a rectilinear polygon P is a rectangular partitioning obtained by the following recursive process. First, the polygon is cut into two subpolygons with an axis-parallel segment contained in $\text{int}(P)$, and then the two subpolygons are partitioned recursively in the same way. A BSP is *anchored* if each of its cuts is anchored.

For so-called histograms we can show that there is always an optimal partition that is an anchored BSP. In fact, we show that any anchored partition of a histogram is a BSP. A *(vertical) histogram* is a rectilinear polygon H that has a horizontal edge seeing every point $q \in \text{int}(H)$. Here we say that an edge e *sees* a point $q \in P$ if there is an axis-parallel segment s connecting e to q that is completely inside $\text{int}(P)$ except possibly its endpoints. We call the horizontal edge that sees all points in the histogram the *base* of the histogram and denote it by $\text{base}(H)$.

Lemma 2 *Any anchored partition of a histogram is a* BSP.

2.1 A 3-approximation algorithm

We present a 3-approximation algorithm for the problem of finding an optimal rectangular partition. First we split P into a set of histograms such that any axis-parallel segment inside P stabs at most three histograms. This can be done in $O(n)$ time [5]. Then, we compute an optimal rectangular partition for each resulting histogram. By proving that this can be done in polynomial time we will have the following result.

Theorem 3 *Let P be a rectilinear simple polygon with n vertices. Then we can compute a rectangular partition of P with stabbing number at most $3 \cdot \text{OPT}$ in polynomial time, where* OPT *is the minimum stabbing number of any rectangular partition of P.*

Let H be a histogram. With a slight abuse of notation, we use n to denote the number of vertices of H. We assume without loss of generality that H is a vertical histogram lying above its base. By Lemmas 1 and 2, H admits an optimal partition that is an anchored BSP. We need the following properties.

Lemma 4 *There is an optimal partition R_{opt} for H that is an anchored* BSP *and such that for every rectangle $r \in R_{\text{opt}}$ we have*

(i) the bottom edge of r is contained in either the top edge of a single rectangle $r' \in R_{opt}$ or in $\text{base}(H)$, and
(ii) the top edge of r contains an edge of H.

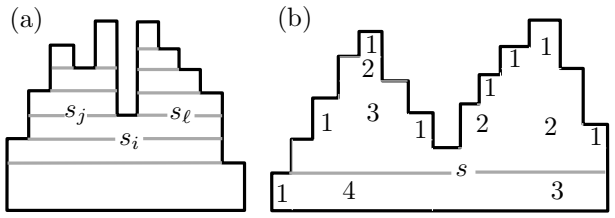


Figure 1: (a) Partitioning a histogram using canonical chords. (b) A partition with a unimodal labeling. The label sequence of the chord s is $4, 3$ and the label sequence of the base is $1, 4, 3$.

In the sequel all partitions have properties (i) and (ii) from Lemma 4 (but not all are anchored).

Our algorithm will do a binary search for the smallest value k such that H admits a partition with stabbing number k. Since there is always a partition with stabbing number at most $2 \log_2 n$ [2], the binary search needs $O(\log \log n)$ steps. It remains to describe our decision algorithm HistogramPartition(H, k), which decides whether H has a partition with stabbing number at most k.

Canonical chords. A *chord* of H is a maximal horizontal segment contained in the interior of H except for its endpoints. A chord s partitions H into two parts. The part above s is a histogram, denoted by $H(s)$. Any partition R of H induces a partition of $H(s)$, denoted by $R(s)$. Now consider a partition of H obtained by adding a chord from each vertex of H for which this is possible. We call the resulting set of chords the *canonical chords* of H (see Figure 1(a)). We treat $\text{base}(H)$ as a canonical chord.

The basic idea behind the algorithm is to treat the chords from top to bottom. Now consider a chord s_i with, say, two chords s_j and s_ℓ immediately above it. Here we say that a chord s_i is *immediately above* another chord s_j, if we can connect s_i to s_j with a vertical segment that does not cross any other chord. One may hope that if we have optimal partitions for $H(s_j)$ and $H(s_\ell)$ then we can somehow "extend" these to an optimal partition for $H(s_i)$. Unfortunately this is not the case, since an optimal partition need not be composed of optimal subpartitions. The next idea is to compute all possible partitions for $H(s_i)$. These can be obtained by considering all combinations of a possible partition for $H(s_j)$ and a possible partition for $H(s_\ell)$. Implementing this idea naively would lead to an exponential-time algorithm, however. Next we show how to compute a subset of all possible partitions that has only polynomial size and is still guaranteed to contain an optimal partition.

Labeled partitions and label sequences. We first introduce some notation and terminology. Let R be any partition of H (having the properties (i) and (ii) in Lemma 4). We say that a rectangle $r \in R$ is *on*

top of a rectangle $r' \in R$ if the bottom edge of r is contained in the top edge of r'. When the bottom edge of r is contained in base(H) then r is on top of the base. A *labeling* of R assigns a positive integer label $\lambda(r)$ to each rectangle $r \in R$. We say that a labeled partition is *valid* (with respect to the stabbing number k) if it satisfies these conditions:

- if r is on top of r' then $\lambda(r) < \lambda(r')$;
- the vertical stabbing number of R equals the maximum label of any rectangle $r \in R$;
- the stabbing number of R is at most k.

Observe that the first two conditions together imply that the stabbing number of R is equal to the maximum label assigned to any rectangle on top of base(H). Also note that any partition with stabbing number k has a valid labeling: for example, one can set $\lambda(r)$ to be equal to the maximum number of rectangles that can be stabbed by a vertical segment whose lower endpoint lies inside r. We will use the labelings to decide which partitions can be ignored and which ones we need to keep.

For a chord s of H, we define the *label sequence of* s with respect to a labeled partition R as the sequence of labels of the rectangles crossed by s, ordered from left to right; here we say that s crosses a rectangle r if s intersects int(r) or the bottom edge of r. We denote this sequence by $\Sigma(s, R)$; see Figure 1(b). A label sequence is *valid* if it consists of at most k labels and the maximum label is at most k. Note that a labeled partition is valid if and only if the label sequence of each of its canonical chords is valid. A label sequence $\lambda_1, \ldots, \lambda_t$ is called *unimodal* if there is an index i such that $\lambda_1 \leqslant \cdots \leqslant \lambda_i$ and $\lambda_i \geqslant \cdots \geqslant \lambda_t$. A labeling of a partition is unimodal if the label sequence of any chord is unimodal. A label sequence can be made unimodal using the following procedure.

MakeUnimodal(Σ)
Let $\Sigma = \lambda_1, \ldots, \lambda_t$, and let λ_{i^*} be a maximum label in the sequence. For each $i < i^*$ set $\lambda_i := \max_{j \leqslant i} \lambda_j$, and for each $i > i^*$ set $\lambda_i := \max_{j \geqslant i} \lambda_j$.

The next lemma states that we can make the label sequences of all canonical chords unimodal, and still keep a valid sequence.

Lemma 5 *Any anchored partition of H of stabbing number at most k admits a valid unimodal labeling.*

Dominated and non-dominated sequences. Next we explain how the labelings help us decide which partitions can safely be discarded. Consider an algorithm that handles the chords from top to bottom, and suppose that the algorithm reaches a chord s. Let R_1 and R_2 be two labeled partitions of $H(s)$. Suppose that $\Sigma(s, R_1)$ is a subsequence of $\Sigma(s, R_2)$. Then there is no need to keep R_2: both partitions have stabbing number at most k so far, and if we can complete R_2 to a partition with stabbing number k of the full histogram H then we can do so with R_1 as well. As another example in which we can disregard one of the two partitions, suppose that $\Sigma(s, R_1) = 1, 1, 3, 1$ and $\Sigma(s, R_2) = 1, 2, 3, 1$, and let $r_1, \ldots, r_4$ be the four rectangle in R_1 reaching the chord s. Then we could have *merged* r_1 and r_2 just before reaching s, that is, we could have terminated r_1 and r_2 and start a new rectangle with label "2"—see Figure 2. The new subsequence is then 2,3,1. This is a subsequence of $\Sigma(s, R_2)$, so we can disregard R_2.

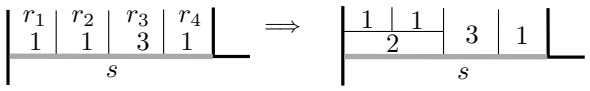


Figure 2: Merging two rectangles.

To make this idea formal, we say that $\Sigma(s, R_1)$ *dominates* $\Sigma(s, R_2)$ if we can obtain a sequence $\Sigma(s, R^*)$ that is a subsequence of $\Sigma(s, R_2)$ by applying the following operation zero or more times to $\Sigma(s, R_1)$:

- Replace a subsequence $\lambda_i, \ldots, \lambda_j$ of $\Sigma(s, R_1)$ by the single label "$\max(\lambda_i, \ldots, \lambda_j) + 1$". Note that we can have $i = j$; in this case the operation just adds 1 to the label λ_i.

Intuitively, if a label sequence dominates another label sequence, then the first sequence has postponed some merging operations that we can still do later on. Thus there is no need to maintain partitions with dominated label sequences. (Note that postponing a merging operation implies that the resulting partition may not be anchored. This means that in the algorithm presented below, we do not restrict ourselves to anchored partitions.) The next lemma gives a bound on the number of label sequences in the worst case.

Lemma 6 *Let $\mathcal{S}$ be any collection of valid unimodal sequences such that no sequence in $\mathcal{S}$ dominates any other sequence in $\mathcal{S}$. Then $|\mathcal{S}| = O(2^{3k/2}/\sqrt{k})$.*

The algorithm. Our decision algorithm is as follows.

HistogramPartition(H, k)

1. Compute the set of canonical chords of H and sort the chords by decreasing y-coordinate.
2. For each chord s in order, compute a collection $\mathcal{R}(s)$ of labeled partitions of $H(s)$, as follows.
   (i) If $H(s)$ is a rectangle then set $\mathcal{R}(s) := \{H(s)\}$.
   (ii) Otherwise s has one or more chords $s_1, \ldots, s_t$ immediately above it—see Figure 3. Find all valid unimodal partitions of $H(s)$ that can be obtained from any combination of partitions in $\mathcal{R}(s_1), \ldots, \mathcal{R}(s_t)$ and whose label sequence is not dominated by the label sequence of any other such partition. (This will be explained below.)

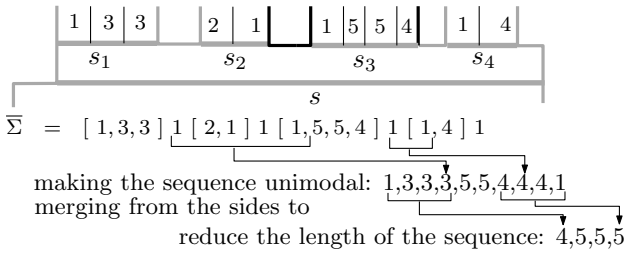


Figure 3: A chord s, the chords immediately above it, and the label sequence $\overline{\Sigma}$ defined by the label sequences of the chords.

Let $\mathcal{R}(s)$ be the set of all computed partitions. If $\mathcal{R}(s)$ is empty, then report that no partition with stabbing number k exists for H, and exit.

3. Return any partition in $\mathcal{R}(\text{base}(H))$.

Next we explain how Step 2(ii) is performed. We assume that $t > 1$, that is, that s has several chords immediately above it; the case $t = 1$ can be handled in a similar (but much simpler) way. In the sequel, we identify each partition with its label sequence and only talk about label sequences. Note that the operations we perform on the label sequences can be easily converted into the corresponding operations on the partitions. For every pair of label partitions $R_1 \in \mathcal{R}(s_1), R_t \in \mathcal{R}(s_t)$ we proceed as follows.

(a) For each $1 < i < t$, consider the set $\mathcal{R}(s_i)$. Note that the label sequences in $\mathcal{R}(s_i)$ all have the same maximum value, M_i. This is because a label sequence dominates any sequence with larger maximum value. (The number of times the maximum label occurs can differ by at most one.) We pick an arbitrary label sequence $\Sigma_i \in \mathcal{R}(s_i)$ for which M_i occurs the minimum number of times.

(b) We now have, besides the partitions $\Sigma_1 \in \mathcal{R}(s_1)$ and $\Sigma_t \in \mathcal{R}(s_t)$, picked a partition Σ_i from each $\mathcal{R}(s_i)$ with $1 < i < t$. Let $\overline{\Sigma}$ be the label sequence obtained by concatenating the sequences Σ_i in order, inserting a label "1" for any horizontal histogram edge incident to a chord s_i, see Figure 3. The labels "1" correspond to new rectangles whose top edge is the given histogram edge. We then make $\overline{\Sigma}$ unimodal. This is done using a variant of the procedure MakeUnimodal explained earlier: the difference is that if we give several consecutive labels the same value, then we merge them into a single new label—see Figure 3.

(c) If the number of labels in $\overline{\Sigma}$ is at most k, then we put $\overline{\Sigma}$ into $\mathcal{R}(s)$. Otherwise $\overline{\Sigma}$ is invalid because it contains too many labels, and we have to merge some rectangles. This is done as follows. Suppose that $\overline{\Sigma}$ contains $k+x$ labels $\lambda_1, \ldots, \lambda_{k+x}$. Then we have to get rid of x labels by merging. Let $x_{\text{left}}, x_{\text{right}}$ be integers such that $x_{\text{left}} + x_{\text{right}} = x + 2$ and both $x_{\text{left}}, x_{\text{right}}$ are non-zero, or $x_{\text{left}} + x_{\text{right}} = x + 1$ and one of $x_{\text{left}}, x_{\text{right}}$ is zero. We merge x_{left} labels from the left into one new label, and x_{right} labels from the right into one, as in Figure 3. In other words, on the left side we replace $\lambda_1, \ldots, \lambda_{x_{\text{left}}}$ by a single new label $\lambda_{x_{\text{left}}}+1$ (and similarly on the right). If there are some labels immediately to the right of $\lambda_{x_{\text{left}}}$ with the same value as $\lambda_{x_{\text{left}}}$, then we include them into the merging process. (We can do this for free, since it reduces the number of labels, without increasing the value of the new label.) If this merging process yields a new label whose value is more than the previous maximum label value, then we simply merge the entire sequence into a single new label. If the value of this label is $k+1$, then we discard the sequence.

After applying the above steps to every pair $R_1 \in \mathcal{R}(s_1), R_t \in \mathcal{R}(s_t)$, we remove from $\mathcal{R}(s)$ all partitions with a label sequence that is dominated by the sequence of some other partition. We do this by comparing every pair of partitions, and checking in $O(k)$ time whether one dominates the other one. After handling s, and for any partition of $H(s)$ denoted by $R^*(s)$, the following lemma shows that the set $\mathcal{R}(s)$ contains at least a partition dominating $\Sigma(s, R^*)$.

Lemma 7 HistogramPartition(H, k) *returns a partition of H with stabbing number at most k if exists.*

The next lemma follows from Lemma 6, and the fact that $k \leqslant 2\log_2 n$.

Lemma 8 HistogramPartition *runs in polynomial time.*

References

[1] P.K. Agarwal, B. Aronov, and S. Suri. Stabbing triangulations by lines in 3D. *Proc. of 11th Symp. on Comp. Geom.* 267–276 (1995).

[2] M. de Berg and M. van Kreveld. Rectilinear decompositions with low stabbing number. *Inf. Process. Lett.* 4:215–221 (1994).

[3] B. Chazelle, H. Edelsbrunner, and L. J. Guibas, The complexity of cutting complexes. *Discr. Comput. Geom.* 6:139–181 (1989).

[4] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: shoot a ray, take a walk. *J. Alg.* 18:403-431 (1995).

[5] C. Levcopoulos. *Heuristics for minimum decompositions of polygons.* Linkoping Studies in Science and Technology, Dissertations, No. 55, 1987.

[6] C.D. Tóth. Axis-aligned subdivisions with low stabbing numbers. *SIAM J. Discrete Math.* 22:1187–1204 (2008).

Coloring Planar Homothets and Three-Dimensional Hypergraphs

Jean Cardinal* Matias Korman*

Abstract

We prove that every set of homothetic copies of a given convex body in the plane can be colored with four colors so that any point covered by at least two copies is covered by two copies with distinct colors. This generalizes a previous result from Smorodinsky [8]. We also show a relation between our proof and Schnyder's characterization of planar graphs. Using this characterization we generalize the first result and show that $k \geq 2$, every three-dimensional hypergraph can be colored with $6(k-1)$ colors so that every hyperedge e contains $\min\{|e|, k\}$ vertices with mutually distinct colors. Furthermore, we also show that at least $2k$ colors might be necessary. This refines a previous result from Aloupis et al. [1].

1 Introduction

The well-known graph coloring problem has several natural generalizations to set systems, or hypergraphs. A proper coloring of a hypergraph can be defined such that no hyperedge is monochromatic, or such that every hyperedge contains some minimum number of distinct colors, for instance. A rich literature exists on these topics; in particular, the two-colorability of hypergraphs (also known as property B), has been well-studied since the sixties [4].

In this paper, we concentrate on coloring geometric hypergraphs, defined by simple objects in the plane. Those hypergraphs serve as models for wireless sensor networks, and associated coloring problems have been investigated recently. Smorodinsky [8] investigated the chromatic number of such geometric hypergraphs, defined as the minimum number of colors required to make every hyperedge non-monochromatic. He considered hypergraphs induced by a collection S of regions in the plane, whose vertex set is S, and the hyperedges are all subsets $S' \subseteq S$ for which there exists a point p such that $S' = \{R \in S : p \in R\}$ (i.e., the regions that contain p). He proved the following result.

Theorem 1 • *Any hypergraph that is induced by a family of n simple Jordan regions such that the union complexity of any m of them is given by $u(m)$ and $u(m)/m$ is non-decreasing is $O(u(n)/n)$-colorable. In particular, any finite family of pseudodisks can be colored with a constant number of colors.*

• *Any hypergraph induced by a finite family of disks is four-colorable*

Later, Aloupis, *et. al.* [1] considered parameterized chromatic numbers. In particular, they studied the quantity $c(k)$, defined as the minimum number of colors required to color a given hypergraph, such that every (sufficiently large) hyperedge has at least k vertices with k distinct colors. This generalizes the previous notion of chromatic number, which corresponds to the case $k = 2$. They proved the following.

Theorem 2 *Any finite family of pseudo-disks in the plane can be colored with $O(k)$ colors in such a way that any point covered by r pseudo-disks is covered by $\min\{r, k\}$ pseudo-disks with distinct colors. For the special case of disks, the number of colors is at most $24k + 1$.*

Our results First, we show in section 3 that Smorodinsky's result for disks holds for every convex body. The proof is similar to that of Smorodinsky, but the graph is constructed in a different way, reminiscent from Schnyder's characterization of planar graphs. This characterization is closely related to the concept of *dimension* of graphs and hypergraphs. Thus, in section 4 we will show the connection between both results and study the chromatic number for three-dimensional hypergraphs. Among other results, we will show that this number is at most $6(k-1)$. This improves the constant of Theorem 2 for this class of hypergraphs, which includes in particular hypergraphs induced by homothets of a triangle. In section 5, we will show a lower bound for all the above problems.

Due to length constraints, some of the proofs have been omitted and/or simplified. A longer version of this paper can be found in [2].

Definitions and notations We consider hypergraphs defined by *ranges*, which are open convex bodies of the form $Q \subset \mathbb{R}^2$ containing the origin. The *scaling* of Q by a factor $\lambda \in \mathbb{R}^+$ is the set $\{\lambda x : x \in Q\}$. The *translate* of Q by a vector $t \in \mathbb{R}^2$ is the set $\{x + t : x \in Q\}$. The *homothet* of Q of *center* t and *scaling* λ is the set $\{\lambda x + t : x \in Q\}$.

*Department of Computer Science, Université Libre de Bruxelles (ULB), {jcardin,mkormanc}@ulb.ac.be

Given a collection S of points in the plane, the *primal hypergraph* defined by these points and a range Q has S as vertex set, and $\{S \cap Q' : Q' \text{ homothet of } Q\}$ as hyperedge set. Similarly, the *dual hypergraph* defined by a set S of homothets of Q has S as vertex set, and the hyperedges are all subsets $S' \subseteq S$ for which there exists a point $p \in \mathbb{R}^2$ such that $S' = \{R \in S : p \in R\}$ (i.e., the set of ranges that contain p).

For a given range Q, the chromatic number $c_Q(k)$ is the minimum number c such that every primal hypergraph (induced by a set of points) can be colored with c colors, so that every hyperedge of size r contains $\min\{r, k\}$ vertices with mutually distinct colors. Similarly, the chromatic number $\bar{c}_Q(k)$ is the smallest number c such that every dual hypergraph (induced by a set of homothets of Q) can be c-colored so that every hyperedge of size r contains $\min\{r, k\}$ vertices with mutually distinct colors.

2 Primal Problem

As a warm-up, we consider the primal version of the problem for $k = 2$. We are interested in giving a coloring of the points of S such that any homothet of Q that contains two or more points of S contains two points of different colors.

Given a set of points S and range Q, the *generalized Delaunay graph* of S induced by Q is a graph $G = (S, E)$ with S as vertex set. For any two points $p, q \in S$, their edge pq is in E if and only there exists a homothet Q' of Q such that Q' contains p, q and no other point of S (in other words $pq \in E \Leftrightarrow Q' \cap S = \{p, q\}$). Note that the Delaunay graph induced by disks corresponds to the classic Delaunay triangulation.

Lemma 3 *[6] For any convex range Q and set of points S, the Delaunay graph of S induced by Q is planar*

The above result has been rediscovered many times along the literature for different types of ranges. See the extended version of this paper [2] for more details.

Theorem 4 *For any convex range Q we have $c_Q(2) \leq 4$.*

Proof. Consider the Delaunay graph of S induced by Q. By Lemma 3, this graph is planar and thus can be four colored. In the following we will show that this coloring is also a valid coloring for our purpose. Let Q' be any homothet of Q containing two or more points of S. Note that if Q' contains exactly two points p and q we have $pq \in E$ by definition of generalized Delaunay graph. In particular, the colors of p and q are different, hence Q' cannot be monochromatic. If Q' has strictly more than two points of S, we continuously shrink it until it contains exactly two points p, q of S (more details of this process can be seen in [2]). As before we have that both $p, q \in Q'$ and $pq \in E$, hence Q' cannot be monochromatic. □

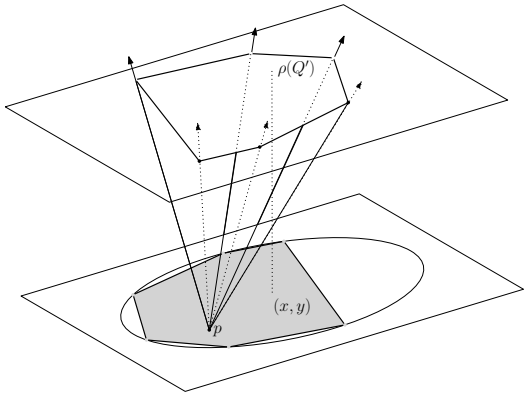


Figure 1: Mapping of a range Q' (in grey) to a point in $\rho(Q') \in \mathbb{R}^3$ and mapping of a point p to a cone. The main property of the mapping is that inclusions are reversed (i.e., a point p is inside Q' if and only if the cone $\pi(p)$ contains $\rho(Q')$).

3 Dual Problem

We now consider the dual version of the problem. That is, we are given a set S of homothets of Q. We say a point is *k-deep* whenever it is covered by at least k elements of S. We apply the same technique as in the primal case: given S, we construct a planar graph $G(S) = (S, E(S))$. The main property of G is that any two-deep point will be covered by two adjacent vertices of $G(S)$.

Let Q' be a homothet of Q with center (x, y) and scaling d. We denote by $\rho(Q')$ the point $(x, y, d) \in \mathbb{R}^3$. Given a set S of homothets of Q, we define $\rho(S) = \{\rho(Q') : Q' \in S\}$.

Similarly, we associate with every point $p = (x, y, d) \in \mathbb{R}^3$ the *cone* $\pi(p)$ defined as follows. Let Q^* be the reflexion of Q about its center. The intersection of $\pi(p)$ with the horizontal plane of height $z \geq d$ is the homothet of Q^* with center (x, y) and scaling $z - d$. The intersection of $\pi(p)$ with a horizontal plane of height $z < d$ is empty. Note that the cone $\pi(p)$ so defined is convex.

We now proceed to define the graph $G(S)$. Its vertex set is S, and two elements Q', Q'' of S are adjacent if and only if there exists a point $p \in \mathbb{R}^3$ such that $\pi(p) \cap \rho(S) = \{\rho(Q'), \rho(Q'')\}$. Thus $G(S)$ is a Delaunay graph in $\mathbb{R}^3$, with cones $\pi(p)$ as ranges.

For any point $p \in \mathbb{R}^2$, let S_p be the set of ranges containing p (i.e., $S_p = \{Q' \in S : p \in Q'\}$). As in the primal case, a valid coloring of G will suffice for our problem:

Lemma 5 *For any $p \in \mathbb{R}^2$ such that $|S_p| \geq 2$, there exist $Q', Q'' \in S_p$ such that $Q'Q'' \in E(S)$.*

Proof. For every point $p = (x, y) \in \mathbb{R}^2$ we consider its cone $\pi((x, y, 0))$. The number of points of $\rho(S)$ contained in this cone is the number of elements of S containing p. We translate vertically upwards $\pi((x, y, 0))$ until it contains exactly two points $\rho(Q')$ and $\rho(Q'')$ (more details of this process can be seen in [2]). These two form an edge in $G(S)$. Since, the translated cone is contained in $\pi(p)$ both Q' and Q'' belong to S_p. □

Lemma 6 *The graph $G(S)$ is planar.*

Proof. By definition of $E(S)$, we know that for every edge $Q'Q'' \in E$ there exists $p \in \mathbb{R}^3$ such that $\pi(p) \cap \rho(S) = \{\rho(Q'), \rho(Q'')\}$. We draw the edge QQ'' as the projection on a horizontal plane of the two line segments connecting respectively $\rho(Q')$ and $\rho(Q'')$ with p.

Note that crossings involving two edges with a common endpoint can be eliminated, so we can simply show that the proposed embedding has no crossing involving vertex-disjoint edges. Consider two such edges uu' and vv', and their corresponding witness cones $\pi_1 \ni u, u'$ and $\pi_2 \ni v, v'$. We must have $u \notin \pi_2$ and $v \notin \pi_1$.

Suppose that the projections of the segments connecting u with the apex of π_1 and v with the apex of π_2 cross at an interior point x. Consider the vertical line ℓ that passes through x: by construction, this line must intersect with both segments at points a and b, respectively. Without loss of generality we assume that a has lower z coordinate than b. From the convexity of π_1, we have $a \in \pi_1$. However, this yields a contradiction, since $v \subseteq \pi(b) \subseteq \pi(a) \subseteq \pi_1$ and we assumed $v \notin \pi_1$. □

Theorem 7 *For any convex range Q we have $\bar{c}_Q(2) \leq 4$.*

4 Coloring Three Dimensional Hypergraphs

The proof of Lemma 6 actually generalizes the "easy" direction of Schnyder's characterization of planar graphs. We first give a brief overview of this fundamental result.

4.1 Poset dimension and Schnyder's theorem

The *vertex-edge incidence poset* of a graph $G = (V, E)$ is a bipartite poset $P = (V \cup E, \preceq_P)$, such that $e \preceq_P v$ if and only if $e \in E$, $v \in V$, and $v \in e$. The *dimension* of a poset $P = (S, \preceq_P)$ is the smallest d such that there exists a injective mapping $f : S \to \mathbb{R}^d$, such that $u \preceq_P v$ if and only if $f(u) \leq f(v)$, where the order $\leq$ is the componentwise partial order on d-dimensional vectors.

Theorem 8 ([7]) *A graph is planar if and only if its vertex-edge incidence poset has dimension at most three.*

The *easy* direction of Schnyder's theorem consists of showing that every graph with vertex-edge incidence poset of dimension at most three is planar. The non-crossing drawing that is considered in the proof is similar to ours, and simply consists, for every edge $e = uv$, of projecting the two line segments $f(e)f(u)$, and $f(e)f(v)$ onto the plane $x + y + z = 0$. This can be identified as a special case of our proof, in which Q is an (equilateral) triangle.

In fact, Lemma 5 directly yields the following corollary.

Corollary 9 *Every hypergraph with vertex-edge incidence poset of dimension at most three is four-colorable.*

4.2 Polychromatic coloring of three-dimensional hypergraphs

We now adapt the above corollary for higher values of k. That is, we are given a three-dimensional hypergraph $G = (V, H)$ and a constant $k \geq 2$. We would like to color the vertices of G such that any hyperedge $e \in H$ contains at least $\min\{|e|, k\}$ vertices with different colors. We will denote by $c_3(k)$ to the minimum number of colors necessary so that any three dimensional hypergraph can be colored. We note that the problem is self-dual: any instance of the dual problem can be transformed into a primal coloring problem by symmetry with respect to the point $(1, 1, 1)$ (assuming that all points are mapped to the interior of the unit cube). Hence, any result for the primal coloring problem will apply to the dual and vice-versa.

In order to avoid degeneracies we assume that no two vertices of G in the mapping share an x, y or z coordinate. For any hyperededge $e \in H$, we define the x-extreme of e as the point $x(e) \in e$ whose mapping has smallest x-coordinate. Analogously we define the y and z-extremes and denote them $y(e)$ and $z(e)$, respectively. For any hyperedgeedge $e \in H$, there exist many points in $\mathbb{R}^3$ that dominate the points of e. We will assume that e is mapped to the point $q_e \in \mathbb{R}^3$ whose x coordinate is equal to the x coordinate of $x(e)$ (analogously for the y and z coordinates). We say that a hyperedge e is *degenerate* if two extremes of e are equal.

Lemma 10 *For any $k \geq 3$, the graph G has at most $3n$ degenerate hyperedges of size exactly k*

For any $2 \leq k \leq n$, we define the graph $G_k(S) = (S, E_k)$, where for any $u, v \in S$ we have $uv \in E_k$ if and only if there exists a point a point $q \in \mathbb{R}^3$ that dominates u, v and at most $k - 2$ other points of S

(that is, we replace hyperedges of size k or less by cliques). The main property of this graph is that any coloring of $G_k(S)$ (in the classic graph coloring sense) induces a polychromatic coloring of G.

We now bound the total number of edges of $G_k(S)$:

Lemma 11 *For any set S of points and $2 \leq k \leq n$, graph $G_k(S)$ has at most $3(k-1)n - 6$ edges*

The bound on the number combined with the minimum degree coloring technique [3] allows us to obtain a coloring of G:

Theorem 12 *For any $k \geq 2$, and three-dimensional hypergraph $G = (V, H)$, the vertices of G can be colored with $6(k-1)$ colors so that any hyperedge $e \in H$ contains $\min\{|e|, k\}$ points with distinct colors. In other words $c_3(k) \leq 6k - 6$*

4.3 Coloring triangles

In this section we will give a very simple application of the previous result. It is easy to show that triangle containment posets (that is, posets of inclusion of homothets of a given triangle) have dimension at most 3. Thus the dual hypergraphs induced by collections of triangles have dimension at most 3, and our result applies.

Theorem 13 *Triangle containment posets have dimension at most 3.*

Corollary 14 *For any $k \geq 3$, any set S of homothets of a triangle can be colored with $6(k-1)$ colors so that any point $p \in \mathbb{R}^2$ covered by r homothets is covered by $\min\{r, k\}$ homothets with distinct colors.*

Note that this result extends the result of Theorem 7 (for the case in which Q is a triangle) to larger values of k. Other than being more general, this proof shows some interesting properties.

Consider the primal variant of Corollary 14: we would like to show that a set S of points can be colored with few colors such that any homothet Δ of a fixed triangle will contain $\min\{|\Delta \cap S|, k\}$ points with different colors. Although the problems are clearly similar, it is not easy to see that they are equivalent. However, by Theorem 13, we know that any instance of the dual problem will generate a three-dimensional hypergraph. Since the dual of a three dimensional hypergraph is another three dimensional hypergraph (as mentioned in Section 4.2), we can apply Theorem 12 to both problems.

In the following Section we will show lower bounds for $c_Q(k)$ and $\bar{c}_Q(k)$ for many different ranges (among them the triangle). Since triangle containment posets have dimension at most 3, this will directly give the same lower bounds for $c_3(k)$. That is, $c_3(k) \geq c_\Delta(k)$, where Δ is any triangular range.

5 Lower Bound

In this section we will give a lower bound for $c_Q(k)$. For that we will use the well known concept of normal direction of Q in a point p (that is, the normal of Q at the boundary point p is the unit vector that is orthogonal to the halfplane that passes through p and supports Q, pointing outwards from Q). We say that a range has m distinct normal directions if there exist m different points such that for any two points, their normals are linearly independent. Note that any affine transformation of a square has two normal directions, a triangle three and a circle has infinitely many.

Lemma 15 *Any range Q with at least three distinct normal directions satisfies $c_Q(k) \geq 4\lfloor k/2 \rfloor$ and $\bar{c}_Q(k) \geq 4\lfloor k/2 \rfloor$.*

The Lemma shows that the upper bounds of Sections 2 and 3 are tight for any range with at three distinct normal directions. Notice that the only bounded shape that does not have three distinct normal directions is the square (and any affine transformation). The above reasonings can be adapted for this case, but for a weaker $c_Q(k) \geq 3\lfloor k/2 \rfloor$ lower bound.

Acknowledgments

The authors would like to thank Prosenjit Bose, Sébastien Collette, Samuel Fiorini and Gwenaël Joret for interesting discussions on the topic.

References

[1] G. Aloupis, J. Cardinal, S. Collette, S. Langerman, and S. Smorodinsky. Coloring geometric range spaces. *Discrete & Computational Geometry*, 41(2):348–362, 2009.

[2] J. Cardinal and M. Korman. Coloring planar homothets and three-dimensional hypergraphs. *CoRR*, abs/1101.0565, 2010.

[3] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, third edition, 2005.

[4] P. Erdős. On a combinatorial problem. *Nordisk Mat. Tidskr.*, 11:5–10, 1963.

[5] E. Horev, R. Krakovski, and S. Smorodinsky. Conflict-free coloring made stronger. In *SWAT*, pages 105–117, 2010.

[6] D. Sarioz. Generalized delaunay graphs with respect to any convex set are plane graphs. *CoRR*, abs/1012.4881, 2010.

[7] W. Schnyder. Planar graphs and poset dimension. *Order*, 5:323–343, 1989.

[8] S. Smorodinsky. On the chromatic number of some geometric hypergraphs. *SIAM Journal on Discrete Mathematics*, 21(3):676–687, 2007.

Flow on Noisy Terrains: An Experimental Evaluation

Mark de Berg Herman Haverkort Constantinos P. Tsirogiannis*

Abstract

We investigate experimentally the changes of drainage structures on triangulated terrains when noise is applied. Given real world data sets, we examine two techniques for identifying watersheds on terrains created using the same data set yet under multiple noise instances. For the needs of our experiments we have developed a robust implementation of an algorithm that computes drainage structures on triangulated terrains under the assumption that water always follows the direction of steepest descent. We provide here also a description of our implementation which is based on CGAL.

1 Introduction

Analysing the flow of water on drainage basins is important for flood prediction, understanding landscape erosion, and other forms of hydrological analysis. To be able to automate such analyses, one needs a computer model of the terrain forming the drainage basin under study and a model for the way in which water flows on the terrain. There are two main terrain models: the digital elevation model (DEM), which is based on a regular grid, and the triangulated irregular network (TIN), which is a 2D triangulation where every vertex is associated with a height value.

The most natural model for water flow is that water follows the direction of steepest descent (DSD) on a surface. As water flows on the surface of a terrain $\mathcal{T}$ following the DSD it accumulates on the local minima of $\mathcal{T}$. For a local minimum p on $\mathcal{T}$, the *watershed* of p is the set of all points on $\mathcal{T}$ from which water flows down to p as it follows the DSD. Since a DEM is a discrete, non-continuous surface, it is not completely clear how to translate the DSD model to DEMs. For TINs, on the other hand, the DSD model can be directly applied (although here one also has to decide how to define flow on flat areas). Nevertheless, existing software for flow modeling often makes use of DEMs, because of the ease of implementing algorithms on DEMs. Implementing the DSD model on a TIN is non-trivial because of robustness problems; indeed, existing software packages for flow computation on TINs do not follow the exact DSD model, but discretise the flow (e.g. by only allowing water to move between a fixed set of points like the vertices of the TIN or the barycenters of the triangles). This poses the question of how reliable the output of such software packages is since they approximate water flow in a coarse way. The only exception is the implementation of Liu *et al.* [8]. However, their software uses fixed precision arithmetic which leads to round-off errors during computation. As they conclude, this eventually produces inconsistencies in the output.

Moreover, an important factor that affects the output of a flow model is the noise that appears in the coordinates of the point set used by a terrain model. Noise comes as a result of sampling with limited accuracy equipment, data conversion between different terrain representation models and calculations under fixed precision arithmetic. For a given TIN, noise can be practically represented at each vertex as an interval of possible height values while the xy-coordinates of the vertex are considered to be fixed. It is therefore interesting to evaluate to what extent the structure of the watersheds of a TIN change if we perturb the height values of its vertices.

Our results. In the current paper, we present a robust C++ software package to compute drainage structures on TINs. Our implementation follows the formal flow model described by de Berg *et al.* [2] and Yu *et al.* [10] , which is based on the exact DSD model. Our package is based on the Computational Geometry Algorithms Library (CGAL), an open source software library for geometric computations that supports the use of exact arithmetic [4]. Thus it is the first ever implementation of a flow model on TINs that follows strictly the DSD assumption and at the same time uses exact arithmetic.

We have used our implementation on TINs that represent real world geographical areas to perform experiments involving perturbations on the height values of their vertex set. We assess which percentage of the area of such a TIN is covered by watersheds of shallow local minima and we test how this percentage changes as the size of the perturbation interval grows. We also investigate how much of the area of indvidual watersheds is preserved between different noise instances. For this reason, we examine two techniques for matching watersheds between TINs that are created when different noise instances are applied to the same terrain data set. Such techniques can be as well

*Dept. of Mathematics and Computer Science, Eindhoven University of Technology, mdberg@win.tue.nl, cs.herman@haverkort.net, ctsirogi@win.tue.nl

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

useful for applications of *data conflation*; that is for identifying the same watershed(s) between TINs that were generated from different data sources yet representing the same geographical area.

2 Experiments

2.1 Software Implementation

To conduct our experiments we have developed a software package that computes drainage structures on triangulated terrains. In particular, we have implemented algorithms for computing paths of steepest descent/ascent, river networks [2], watershed maps and strip maps [10] on TINs. Our software was implemented in C++ using CGAL which provided basic geometric objects, predicates and number types as building blocks for our needs.

The implemented algorithms that compute watershed maps and strip maps take as an input argument an object that is a model of the `CGAL::Triangulation_euclidean_traits_xy_3`; this is the appropriate triangulation type in CGAL to represent a terrain. The triangulation object is then converted to an object of an augmented version of the `CGAL::Polyhedron_3` type. This is because boundaries between watersheds and strips may not always coincide with the triangulation edges and can as well extend through triangle interiors. The output of the watershed map algorithm is an object of the augmented `CGAL::Polyhedron_3` type where each facet, edge and vertex on the surface is tagged with the watershed that it belongs to. Also each of the watersheds in the output is provided as a model of the graph concept of the BOOST Graph Library [6].

We have also designed a traits class with the name `Drainage_traits_2` that provides all the predicates and geometric object types that are essential for the main algorithms. This traits class is a template class with two parameters; a version of the `CGAL::Triangulation_euclidean_traits_xy_3` model and a model of a CGAL linear geometric kernel [3]. The linear kernel type parameter may be instatiated with any number type, yet in practice the algorithms of this package will not execute properly unless an arbitrary precision number type is used such as `gmpq`. This is due to the fact that fixed precision arithmetic is not enough to represent the coordinates of the intersection points between a path of steepest descent/ascent and terrain edge interiors [8]. Because of this, we have run our experiments using exclusively as a kernel parameter the `CGAL::Exact_predicates_exact_constructions_kernel`. We intend to show in our future work how the bit-size of the computed structures affects in practice the applicability of the described flow model.

We have ran our implementation in a Linux Ubuntu operating system version 9.10 using the GNU g++ version 4.4 compiler and CGAL version 3.6.1.

2.2 Input Data Specifications and Noise Model

The TINs that we have used for our experiments are constructed from DEM data sets that are publicly available from the U.S. Geological Survey (USGS) server [9]. The data files hosted in this server are of the USGS *DEM* digital format. Each file stores a grid terrain of 1201×1201 cells where the dimension of each cell is 90 meters and the elevation values of the cells are integers.

We have created three TINs, each derived from a different DEM data set. Each of these TINs consists of 50000 vertices and roughly 100000 triangles. The vertex set of each TIN was created by sampling a set of points on the surface of one of the DEMs. A Delaunay triangulation was then constructed on the xy-projection of the selected point set. Thus the final TIN can be seen as the result of lifting the vertices of this triangulation up to their original elevation values. The names of the TIN data sets and the range of elevations among the vertices of each TIN are listed in Table 1.

Name	Elevation Range (in meters)
`aztec`	[1704, 3523]
`marion`	[213, 457]
`carlsbad`	[861,2256]

Table 1: The names and the elevations ranges of the TIN data sets that we use in our experiments.

Perturbations on the elevations of the TIN vertices were conducted in the following way. To the elevation of each TIN vertex we added a value picked uniformly at random from an interval of the form $[-\eta, \eta]$. The value of η used in each experiment is explicitly stated in the description of the experiment.

3 Experiments

3.1 Area covered by small watersheds

In the first experiment we examine how large is the part of the terrain covered by watersheds of spurious minima and how this changes as noise is added to the elevations of the vertices. There exist several criteria in the GIS literature to define whether a local minimum is a spurious minimum or not e.g. topological persistence [5]. In our experiment we consider a local minimum to be spurious if the xy-projection of its watershed covers less than one thousandth of the xy-projection of the total terrain area. We consider all watersheds that are larger than this threshold as *sufficiently large*. Deciding which threshold value is

appropriate to distinguish which watersheds are sufficiently large is a debatable issue and depends on the scale that is used in the analysis.

We have conducted the experiment in the following way; for each of the original TIN data sets we have constructed six new TINs by perturbing the elevation values of their vertices. We use a perturbation interval $[-\eta, \eta]$ of different size for each new TIN instance by choosing the value of η from the range $\{0.5m, 1.0m, \ldots, 3.0m\}$. We compute the watershed map for each new TIN and we measure:

- the number of watersheds that belong to spurious minima.
- the percentage of the total area that is covered by those watersheds on the xy-projection of the terrain.

The results of this experiment are summarized in Table 2. For the `aztec` data set we see that roughly 50% of the terrain is covered by relatively small watersheds and this percentage reaches approximately 55% as the max size of the absolute perturbation increases to 3 meters. For the `carlsbad` data set this percentage reaches up to 72% . Worse than that, the area covered by small watersheds on the `marion` data set rises dramatically even when the max absolute noise is half a meter. This is due to the fact that the elevation range of this data set is quite small, about 200 meters. Thus the original TIN consists of large flat areas that break into small watersheds even with small perturbations. The number of small watersheds rises substantially for all of the data sets with respect to the maximum noise and in the `marion` more than doubles when the max absolute noise reaches 3 meters.

3.2 Watershed area maintained among different noise instances

Next we investigate to what extent large watersheds on a TIN occupy the same area after perturbations of the terrain vertices. We proceed with the following definition.

Consider a TIN $\mathcal{T}_1$ and let $v_1 \in \mathcal{T}_1$ be a vertex that is also a local minimum. Let $\mathbb{T} = \{\mathcal{T}_2, \mathcal{T}_3, \ldots, \mathcal{T}_m\}$ be a set of TINs such that each $\mathcal{T}_i \in \mathbb{T}$ can be induced by modifying the height values of the vertices of $\mathcal{T}_1$. Assume that for every $\mathcal{T}_i \in \mathbb{T}$ the vertex v_i that corresponds to (has the same xy-coordinates as) v_1 is also a local minimum of $\mathcal{T}_i$. Let w^i_{xy} be the xy-projection of the watershed of v_i for every $1 \leq i \leq m$. We call the *core watershed* of v_1 the intersection $\cap_{1 \leq i \leq m} w^i_{xy}$.

Based on this definition of core watersheds we perform our next experiment as follows. Given one of our TIN data sets we construct three new instances by perturbing the vertices of the original TIN on the z-coordinate. The noise interval that we use to derive each of the three new instances is $[-1.0m, 1.0m]$. We compute the watershed map on the three new TINs as well as on the original TIN. We then distinguish on the original TIN all local minima whose watersheds cover individually more than one thousandth of the terrain xy-area. We then compute the core watersheds of these minima and calculate the percentage of the terrain xy-area that these core watersheds cover.

The results of this experiment (not displayed in this version of the paper) show that only roughly 5% of the total terrain xy-area is indeed covered by the core watersheds. This is partly due to the fact that not all vertices that constitute local minima on the original terrain appear as local minima also on the new terrain instances. Consider a vertex v of locally minimum elevation that has a very small height difference from its neighbours on the triangulation. Then even a slight perturbation among all terrain vertices may cause another nearby vertex to become a local minimum instead of v while no large changes may be induced in the structure of the surrounding drainage area.

Being aware of this issue we revise our experiment by adopting the following approach. Let $\mathbb{T} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4\}$ be the four TINs that we have created, that is the original TIN data set and the three new instances acquired by applying noise on the original TIN. Let $\mathcal{W}_i$ be the set of all sufficiently large watersheds on the surface of $\mathcal{T}_i \in \mathbb{T}$. Let w be a watershed in $\mathcal{W}_i$. Our intention is to match w with exactly one large watershed from *each* other instance $\mathcal{W}_j, j \neq i$ such that the matched watersheds have the largest possible intersection on the xy-plane.

More formally, we consider a hypergraph where each graph-node corresponds to a watershed in $\cup_{1 \leq i \leq 4} \mathcal{W}_i$. Each hyperedge is incident to four nodes, each node corresponding to a watershed from a different TIN instance. Also every hyperedge has a positive weight, equal to the area of the intersection of (the xy-projections of) the watersheds corresponding to the incident nodes. Our goal is to compute the maximum weight 4D-matching on the described hypergraph. This graph problem is known to be NP-hard, yet there exists a 3-approximation algorithm based on local search [1]. We have implemented this local search algorithm for the needs of our experiments and tested it on the described TINs. The results obtained from the local search matching algorithm are listed on Table 3.

Data set	# of Matched Watersheds	Area %
`aztec`	203	22.11
`marion`	77	5.26
`carlsbad`	174	15.24

Table 3: Results of matching watersheds using the 4D matching approximation algorithm.

Data set: Aztec			
Noise Interval (in meters)	# of Small Watersheds	# of All Watersheds	Total Area of Small Watersheds %
$[0, 0]$	2032	2348	48.02
$[-0.5, 0.5]$	1735	2061	48.67
$[-1.0, 1.0]$	1797	2119	49.20
$[-1.5, 1.5]$	1919	2230	52.85
$[-2.0, 2.0]$	1984	2292	53.54
$[-2.5, 2.5]$	2071	2372	56.45
$[-3.0, 3.0]$	2089	2395	55.62

Data set: Marion			
Noise Interval (in meters)	# of Small Watersheds	# of All Watersheds	Total Area of Small Watersheds %
$[0, 0]$	2041	2233	38.80
$[-0.5, 0.5]$	2972	3176	68.23
$[-1.0, 1.0]$	3301	3509	71.60
$[-1.5, 1.5]$	3570	3755	75.82
$[-2.0, 2.0]$	3805	3972	79.34
$[-2.5, 2.5]$	4069	4206	83.37
$[-3.0, 3.0]$	4350	4458	86.95

Data set: Carlsbad			
Noise Interval (in meters)	# of Small Watersheds	# of All Watersheds	Total Area of Small Watersheds %
$[0, 0]$	2032	2787	59.38
$[-0.5, 0.5]$	1915	2212	52.72
$[-1.0, 1.0]$	2085	2374	56.43
$[-1.5, 1.5]$	2315	2594	59.86
$[-2.0, 2.0]$	2444	2843	64.21
$[-2.5, 2.5]$	2862	3096	68.17
$[-3.0, 3.0]$	3058	3267	72.83

Table 2: The number of small watersheds and percentage of xy-area they cover on the TIN data sets

4 Conclusions and Future work

As expected, a large part of the area in the acquired data sets is covered by small watersheds. We further show that the application of small perturbations magnifies substantially this phenomenon especially on terrains with relatively small elevation range. Sufficiently large watersheds maintain only a small part of their total area when applying small perturbations on the TIN vertices. We intend to evaluate how much these results can improve with the use of a technique that floods spurious local minima. We also opt to conduct experiments with data sets of different resolution and compare the output with the current results.

References

[1] E.M. Arkin and R. Hassin On Local Search for Weighted k-Set Packing. *Mathematics of Operations Research*, 23:640–648, 1998.

[2] M. de Berg, P. Bose, K. Dobrint, M. van Kreveld, M. Overmars, M. de Groot, T. Roos, J. Snoeyink and S. Yu. The Complexity of Rivers in Triangulated Terrains. In *Proc. 8th Canadian Conference on Computational Geometry*, pages 325–330, 1996.

[3] H. Brönnimann, A. Fabri, G.J. Giezeman, S. Hert, M. Hoffmann, L. Kettner, S. Pion and S. Schirra. 2D and 3D Geometry Kernel. In *CGAL User and Reference Manual*, CGAL Editorial Board, 3.7 edition, 2010.

[4] CGAL, Computational Geometry Algorithms Library. `http://www.cgal.org`.

[5] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological Persistence and Simplification. In *Proc. 41st IEEE Symposium on Foundations of Computer Science*, pages 45–463, 2000.

[6] A. Fabri, F. Cacciola and R. Wein. CGAL and the Boost Graph Library. In *CGAL User and Reference Manual*, CGAL Editorial Board, 3.7 edition, 2010.

[7] M. McAllister. A Watershed Algorithm for Triangulated Terrains. In *Proc. 11th Canadian Conference on Computational Geometry*, 103–106, 1999.

[8] Y. Liu and J.Snoeyink. Flooding Triangulated Terrain. In *Proc. 11th International Symposium on Spatial Data Handling*, pages 137–148, 2005.

[9] United States Geological Survey file server. `http://dds.cr.usgs.gov/pub/data/DEM/250/`.

[10] S. Yu, M. van Kreveld and J. Snoeyink. Drainage Queries in TINs: From Local to Global and Back Again. In *Proc. 7th International Symposium on Spatial Data Handling*, pages 13–1, 1996.

4-Holes in Point Sets

Oswin Aichholzer* Ruy Fabila-Monroy† Hernán González-Aguilar ‡ Thomas Hackl*
Marco A. Heredia § Clemens Huemer¶ Jorge Urrutia‡ Birgit Vogtenhuber*

Abstract

We consider a variant of a question of Erdős on the number of empty k-gons (k-holes) in a set of n points in the plane, where we allow the k-gons to be non-convex. We show bounds and structural results on maximizing and minimizing the number of general 4-holes, and maximizing the number of non-convex 4-holes.

1 Introduction

Let S be a set of n points in general position in the plane. A k-gon is a simple polygon spanned by k points of S. A *k-hole* is an empty k-gon, that is, a k-gon which does not contain any points of S in its interior.

In 1978 Erdős [5] raised the following question for convex k-holes: "What is the smallest integer $h(k)$ such that any set of $h(k)$ points in the plane contains at least one convex k-hole?" As already observed by Esther Klein, every set of 5 points determines a convex 4-hole, and 10 points always contain a convex 5-hole, a fact proved by Harborth [8]. However, in 1983 Horton showed that there exist arbitrarily large sets of points not containing any convex 7-hole [9]. It again took almost a quarter of a century after Horton's construction to answer the existence question for 6-holes. In 2007/08 Nicolás [11] and independently Gerken [7] proved that every sufficiently large point set contains a convex 6-hole.

Erdős also proposed the following variation of the problem [6]. "What is the least number $h_k(n)$ of convex k-holes determined by any set of n points in the plane?" We know by Horton's construction that $h_k(n) = 0$ for $k \geq 7$. For $k \leq 6$, upper and lower bounds on $h_k(n)$ exist; see [1] for a survey.

*Institute for Software Technology, University of Technology, Graz, Austria, [oaich|thackl|bvogt]@ist.tugraz.at

†Departamento de Matemáticas, Cinvestav, México, ruyfabila@math.cinvestav.edu.mx

‡Instituto de Matemáticas, Universidad Nacional Autónoma de México, Mexico City, Mexico, [hernan|urrutia]@matem.unam.mx

§Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, D.F. México, marco@ciencias.unam.mx

¶Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, Barcelona, Spain, clemens.huemer@upc.edu

In this paper we generalize the latter problem by allowing a k-hole to be non-convex. Thus, whenever we refer to a k-hole, it might be convex or non-convex, and we will explicitly state it when we restrict investigations to one of these two classes.

Note that a set of four points in non-convex position might span up to three 4-holes; that is, the number of k-holes can be larger than $\binom{n}{k}$, the maximum number of convex k-holes.

We first investigate sets of small cardinality (Section 2), and then consider the following tasks: maximizing the number of 4-holes (Section 3), maximizing the number of non-convex 4-holes (Section 4), and minimizing the number of 4-holes (Section 5). In addition to the best possible lower and upper bounds on their number, we also show which families of point sets obtain these bounds.

2 Small Sets

Even to determine the number of small holes is surprisingly intriguing. For $n \leq 11$, Table 1 shows the minimum number of convex 4-holes, the maximum number of non-convex 4-holes, the minimum and maximum number of (general) 4-holes, and, for easy comparison, the number of 4-tuples.

n	convex min	non-convex max	general min	general max	$\binom{n}{4}$
4	0	3	1	3	1
5	1	8	5	9	5
6	3	18	15	22	15
7	6	36	35	43	35
8	10	64	66	77	70
9	15	100	102	126	126
10	23	150	147	210	210
11	32	216	203	330	330

Table 1: Number of 4-holes for $n = 4, \ldots, 11$ points [1].

Obviously, the maximum number of convex 4-holes is $\binom{n}{4}$, obtained by sets in convex position. For minimizing the number of convex 4-holes, the currently best known bounds are $\frac{n^2}{2} - O(n) \leq h_4(n) \leq 1.9397n^2 + o(n^2)$, see [3, 4, 12].

For $n = 4, \ldots, 7$ it can be seen from Table 1 that the minimum number of 4-holes is $\binom{n}{4}$. In contrast, $\binom{n}{4}$ is the maximum number of 4-holes for $n = 9, \ldots, 11$, so the structure of extremal sets seems to switch.

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

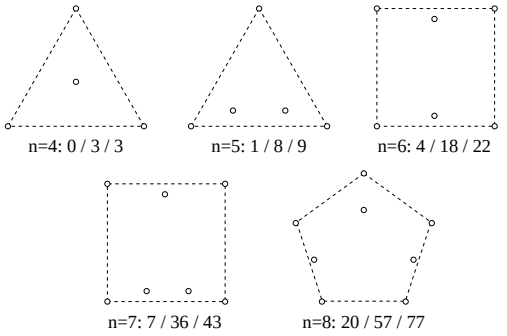


Figure 1: Point sets maximizing the number of 4-holes for $n = 4, \ldots, 8$. Shown are the number of convex, non-convex, and general 4-holes.

Figure 1 shows point sets maximizing the number of 4-holes for $n = 4, \ldots, 8$. The results for $n > 8$ suggest that sets in convex position might maximize the number of 4-holes for $n \geq 9$. Indeed, this will be the first result we prove for general 4-holes (Section 3).

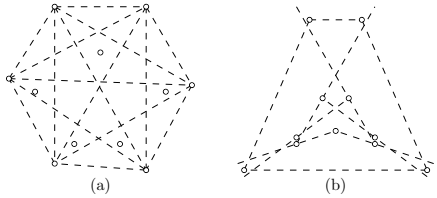


Figure 2: Two unique extremal sets for $n = 11$ points: (a) maximizes the number of non-convex 4-holes, and (b) minimizes the number of 4-holes.

Figure 2 shows two extremal sets for $n = 11$ points. Each set represents the unique order type which reaches the extreme value. The left set maximizes the number of non-convex 4-holes, namely 216, and consists of a convex 5-hole inside a convex 6-gon. The total number of 4-holes in this set is 267; i.e., it contains in addition 51 convex 4-holes. The set on the right side minimizes the number of general 4-holes. It contains 51 convex and 152 non-convex 4-holes, thus in total the minimum of 203 4-holes.

3 Maximizing the Number of (General) 4-Holes

Lemma 1 *Let Δ be a non-empty triangle in S. There are at most three non-convex 4-holes spanned by the three vertices of Δ plus a point of S in the interior of Δ.*

Proof. Let p_1, p_2, and p_3 be the vertices of Δ. Observe that any non-convex 4-hole has to use two edges of Δ. Thus there are three choices for the unused edge of Δ, and for each choice there is at most one way to complete the two used edges of Δ to a 4-hole. Assume to the contrary that two different 4-holes avoid the edge p_2p_3 and use points q_1 and q_2, respectively, in the interior. Then q_2 lies outside the two triangles formed by $p_1q_1p_2$ and $p_1q_1p_3$. Thus q_2 lies in the triangle formed by $p_2q_1p_3$. But then q_1 must lie in one of the triangles spanned by $p_1q_2p_2$ and $p_1q_2p_3$, a contradiction. □

Theorem 2 *For $n \geq 9$ the number of 4-holes is maximized by a set of n points in convex position.*

Proof. In the following we assign every non-convex 4-tuple of points to the three vertices of its convex hull and call this the *representing* triangle of the potential non-convex 4-holes. By Lemma 1, any non-empty triangle represents at most three 4-holes, and any convex 4-tuple gives at most one 4-hole.

Let T be the number of non-empty triangles. As any non-empty triangle induces at least one 4-tuple in non-convex position, we get

$$\binom{n}{4} + 2T \tag{1}$$

as a first upper bound on the number of 4-holes of a point set.

Note that a triangle Δ with $k \geq 1$ interior points is counted $k+2$ times in (1), namely k times in the $\binom{n}{4}$ 4-tuples plus the extra 2 as Δ is non-empty. Thus for $k > 1$ we have over-counted the number of non-convex 4-holes assigned to Δ; cf. Lemma 1. Moreover, many of the convex 4-gons might not be empty and thus no 4-holes. Therefore we now analyze how many of the counted potential 4-holes can be reduced from (1). We will do this by assigning (possibly multiple) markers for over-counted 4-holes to convex 4-tuples and non-empty triangles.

As above, let Δ be a triangle with $k \geq 1$ interior points, and consider all 4-tuples consisting of the three vertices of Δ plus an extra point p. We distinguish two cases.

First let p be one of the $n-k-3$ points outside Δ. If the resulting 4-tuple is convex, we mark this 4-tuple, as it is not empty and thus no 4-hole. If the 4-tuple is non-convex, we mark the triangle which represents the potential non-convex 4-hole, as at least one of the three possible 4-holes of this 4-tuple is non-empty.

In the second case we consider the k points inside Δ. As argued above, Δ was counted $k+2$ times. But by Lemma 1, there are at most three 4-holes using one interior point of Δ and thus represented by Δ. Therefore we assign $k-1$ markers to Δ.

Altogether we have distributed $n-4$ markers while considering Δ. Repeating this for all non-empty triangles, we obtain a total of $(n-4) \cdot T$ markers.

A non-empty convex 4-tuple might have received up to 4 markers in this way, one from each of its subtriangles. That is, we have at most 4 times as many markers as convex 4-tuples which we can reduce from the upper bound (1).

A non-empty triangle Δ with $k \geq 1$ interior points might have got $4 \cdot (k-1)$ markers: For its interior points, Δ received $k-1$ markers from the second case, and for each non-empty triangle formed by two vertices of Δ and one point inside Δ, we received one marker from the first case. As at least three of the considered inner triangles are empty (the ones spanned by an edge e of Δ and the interior point closest to e), the first case gives at most $3 \cdot (k-1)$ additional markers, resulting in a total of at most $4 \cdot (k-1)$ markers for Δ. As Δ was counted $k+2$ times, but represents at most three 4-holes (Lemma 1), we have at most $4 \cdot (k-1)$ markers for at least $(k+2)-3 = k-1$ over-counted objects. Thus, in both cases we over-counted reducible terms at most by a factor of 4. We therefore can reduce the number of potential 4-holes by one quarter of the distributed markers, namely by $\frac{n-4}{4} \cdot T$. This leads to the improved upper bound

$$\binom{n}{4} + 2T - \frac{n-4}{4} \cdot T$$

for the number of 4-holes. For $n \geq 12$ this is at most $\binom{n}{4}$, the number of 4-holes for a set of points in convex position. Together with the results from Table 1 for $n = 9, \ldots, 11$, this proves the theorem. □

4 Maximizing the Number of Non-Convex 4-Holes

Lemma 3 *The number of non-convex 4-holes of any set of n points is at most $\frac{n(n-1)(n-2)}{2} = \frac{n^3}{2} - O(n^2)$.*

Proof. By Lemma 1, any non-empty triangle generates at most three non-convex 4-holes, and there are at most $\binom{n}{3}$ such triangles in a set of n points. □

Theorem 4 *For every $n' > 0$ there exist sets of n points for some $n \in \{n', \ldots, 2n'\}$, with at least $\frac{n^3}{2} - O(n^2 \log n)$ non-convex 4-holes.*

Proof. We consider the special point sets $\mathcal{X}_m$, $m \geq 1$, with $|\mathcal{X}_m| = n = 2^{m+1} - 2$ points, introduced in [10]. The point sets are defined recursively in layers, starting with two points $\mathcal{X}_1 := \mathcal{R}_1$ in the first layer. An additional layer $\mathcal{R}_i$ is added to $\mathcal{X}_{i-1} := \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{i-1}$ by placing two new points close to any point in $\mathcal{R}_{i-1}$ outside the convex hull of $\mathcal{X}_{i-1}$, such that the following conditions hold: (1) $\mathcal{X}_i = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_i$ is in general position, (2) $\mathcal{R}_i$ are the extremal points of $\mathcal{X}_i$, and (3) any triangle determined by $\mathcal{R}_i$ contains precisely one point of $\mathcal{X}_i$ in its interior. See Figure 3 for an example and [10] for a detailed description of the construction. Furthermore, in [10] it is shown that every triangle spanned by $\mathcal{X}_m$ contains at most one interior point of $\mathcal{X}_m$; i.e., every non-empty triangle of $\mathcal{X}_m$ contains exactly one point. Using Lemma 1, the number of non-convex 4-holes of $\mathcal{X}_m$ is three times the number of non-empty triangles.

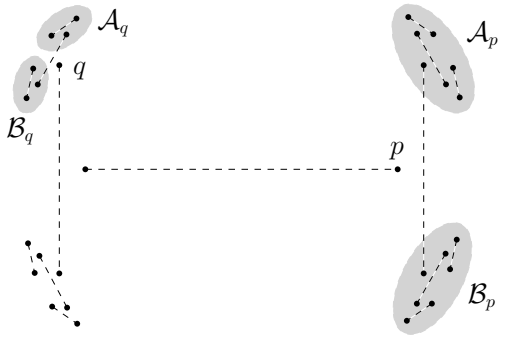


Figure 3: Example for $m = 4$ of the special point set defined in [10].

For each point of the set $\mathcal{X}_m$, we count the number of triangles that contain it. First, fix a point in the first layer $\mathcal{R}_1$, say p in Figure 3. Any triangle containing p is formed by one point of $\mathcal{A}_p$, one point of $\mathcal{B}_p$, and one point of the remaining set $\mathcal{C}_p = \mathcal{X}_m \backslash \{\mathcal{A}_p \cup \mathcal{B}_p \cup \{p\}\}$. We say that $\mathcal{A}_p$ and $\mathcal{B}_p$ are the *induced subsets* of p, and that $\mathcal{C}_p$ is the *remainder* (of $\mathcal{X}_m$) for p. As $a_1 := |\mathcal{A}_p| = |\mathcal{B}_p| = \frac{n-2}{4}$ and $c_1 := |\mathcal{C}_p| = n - 2 \cdot a_1 - 1 = \frac{n}{2}$, this gives $a_1^2 \cdot c_1$ triangles containing p, and thus the number of triangles containing a point of $\mathcal{R}_1$ is $2 \cdot a_1^2 \cdot c_1 = 2 \cdot (\frac{n-2}{4})^2 \cdot \frac{n}{2}$.

Now consider a point q in the second layer $\mathcal{R}_2$. Its induced subsets $\mathcal{A}_q$ and $\mathcal{B}_q$ have size $a_2 = \frac{n-6}{8}$, and the remainder $\mathcal{C}_q$ has $c_2 = n - 2 \cdot a_2 - 1 = \frac{3n+2}{4}$ points. In combination with $r_2 := |\mathcal{R}_2| = 4$ this gives a total of $4 \cdot (\frac{n-6}{8})^2 \cdot \frac{3n+2}{4}$ triangles containing a point of $\mathcal{R}_2$.

In general, $|\mathcal{R}_i| = r_i = 2^i$, and the size of the two induced subsets of any point p_i in $\mathcal{R}_i$ is

$$a_i = \frac{1}{r_{i+1}}(n - |\mathcal{X}_i|) = \frac{n - (2^{i+1} - 2)}{2^{i+1}}.$$

Thus, with the size of the corresponding remainder $\mathcal{C}_{p_i}$ of

$$c_i = n - 2 \cdot a_i - 1 = \frac{(2^i - 1)n + 2^i - 2}{2^i},$$

we get $r_i \cdot a_i^2 \cdot c_i$ triangles containing one point of $\mathcal{R}_i$.

Using that every non-empty triangle of $\mathcal{X}_m$ gives three non-convex 4-holes, and summing up over all layers $\mathcal{R}_i$, we obtain

$$3 \cdot \sum_{i=1}^{m} r_i \cdot a_i^2 \cdot c_i \quad =$$

$$3 \cdot \sum_{i=1}^{m} 2^i \left(\frac{n - (2^{i+1} - 2)}{2^{i+1}} \right)^2 \frac{(2^i - 1)n + 2^i - 2}{2^i} \quad =$$

$$\frac{1}{2} n^3 + \left(\frac{39}{4} - 3 \log_2(n+2) \right) n^2 - O(n \log n)$$

for the total number of non-convex 4-holes of $\mathcal{X}_m$. □

5 Minimizing the Number of (General) 4-Holes

We obtained the following observation for general 4-holes by checking all corresponding point sets from the order type data base [2].

Observation 1 *Let S be a set of $n = 8$ points in the plane in general position, and $p_1, p_2 \in S$ two arbitrary points of S. Then S contains at least five 4-holes having p_1 and p_2 among their vertices.*

Based on this simple observation, we derive the following lower bound for the number of 4-holes.

Theorem 5 *Let S be a set of $n \geq 8$ points in the plane in general position. Then S contains at least $\frac{5}{2}n^2 - O(n)$ 4-holes.*

Proof. We consider the point set S in x-sorted order, $S = \{p_1, \ldots, p_n\}$, and sets $S_{i,j} = \{p_i, \ldots, p_j\} \subseteq S$. The number of sets $S_{i,j}$ having at least 8 points is

$$\sum_{i=1}^{n-7} \sum_{j=i+7}^{n} 1 = \sum_{i=1}^{n-7} n - i - 6 = \frac{n^2}{2} - \frac{13}{2}n + 21.$$

By Observation 1, each set $S_{i,j}$ contains at least five 4-holes having p_i and p_j among their vertices (take the six points of $S_{i,j}$ which are nearest to the segment p_ip_j). Moreover, as p_i and p_j are the left- and rightmost points of $S_{i,j}$, they are also the left- and rightmost points for each of these 4-holes. This implies that any 4-hole of S counts for at most one set $S_{i,j}$, which gives a lower bound of $\frac{5}{2}n^2 - O(n)$ for the number of 4-holes in S. □

Note that there exist sets which contain fewer than $1.94n^2$ convex 4-holes, while by the above result any set contains at least $2.5n^2$ (general) 4-holes.

6 Conclusion

We have shown lower and upper bounds on the numbers of (general) and non-convex 4-holes in point sets.

A natural generalization of this work is to consider similar questions for $k > 4$. For example, we have been able to show that for every constant $k \geq 3$, the maximum number of non-convex k-holes is asymptotically smaller than the maximum number of convex k-holes. This gives rise to the following conjecture, which is a general version of Theorem 2, and part of our ongoing research on this topic.

Conjecture 1 *For any constant $k \geq 5$ and sufficiently large n, the number of k-holes is maximized by a set of n points in convex position.*

Acknowledgments

Research on this topic was initiated during the *Third Workshop on Discrete Geometry and its Applications* in Morelia (Michoacán, Mexico). We thank Edgar Leonel Chávez González and Feliu Sagols for helpful discussions. Research of Oswin Aichholzer, Thomas Hackl, and Birgit Vogtenhuber supported by the FWF [Austrian Fonds zur Förderung der Wissenschaftlichen Forschung] under grant S9205-N12, NFN Industrial Geometry. Research of Clemens Huemer partially supported by projects MEC MTM2009-07242 and Gen. Cat. DGR 2009SGR1040. Research of Marco Antonio Heredia Velasco, Hernán González-Aguilar, and Jorge Urrutia partially supported by CONACyT (Mexico) grant CB-2007/80268.

References

[1] O. Aichholzer. [Empty] [colored] k-gons - Recent results on some Erdös-Szekeres type problems. In *Proc. XIII Encuentros de Geometría Computacional*, pages 43–52, Zaragoza, Spain, 2009.

[2] O. Aichholzer and H. Krasser. The point set order type data base: A collection of applications and results. In *Proc. 13th Canadian Conference on Computational Geometry CCCG'01*, pages 17–20, 2001.

[3] I. Bárány and P. Valtr. Planar point sets with a small number of empty convex polygons. *Studia Sci. Math. Hungar.*, 41(2):243–266, 2004.

[4] A. Dumitrescu. Planar sets with few empty convex polygons. *Studia. Sci. Math. Hungar.*, 36(1–2):93–109, 2000.

[5] P. Erdős. Some more problems on elementary geometry. *Austral. Math. Soc. Gaz.*, 5:52–54, 1978.

[6] P. Erdős. Some old and new problems in combinatorial geometry. *In* Convexity and Graph Theory*, M. Rosenfeld et al., eds., Annals Discrete Math.*, 20:129–136, 1984.

[7] T. Gerken. Empty convex hexagons in planar point sets. *Discrete and Computational Geometry*, 39(1–3):239–272, 2008.

[8] H. Harborth. Konvexe fünfecke in ebenen punktmengen. *Elemente Math.*, 33:116–118, 1978.

[9] J. Horton. Sets with no empty convex 7-gons. *Canad. Math. Bull.*, 26(4):482–484, 1983.

[10] G. Károlyi, J. Pach, and G. Tóth. A modular version of the Erdős-Szekeres theorem. *Studia. Sci. Math. Hungar.*, 38(1–4):245–260, 2001.

[11] C. Nicolás. The empty hexagon theorem. *Discrete and Computational Geometry*, 38(2):389–397, 2007.

[12] P. Valtr. On the minimum number of empty polygons in planar point sets. *Studia. Sci. Math. Hungar.*, 30:155–163, 1995.

Flow Computations on Imprecise Terrains

Anne Driemel* Herman Haverkort† Maarten Löffler‡ Rodrigo I. Silveira§

Abstract

We study the computation of the flow of water on imprecise terrains. We consider two approaches to modeling flow on a terrain: one where water can only flow along the edges of a predefined graph (for example a grid, a triangulation, or its dual), possibly non-planar, and one where water flows across the surface of a polyhedral terrain in the direction of steepest descent. In both cases each vertex has an imprecise height, given by an interval of possible values, while its (x, y)-coordinates are fixed. For the first model, we give a simple $O(n \log n)$ time algorithm to compute the maximal watershed of a vertex, where n is the number of edges of the graph. We show that, in contrast, in the second model the problem of deciding whether one vertex may be contained in the watershed of another is NP-hard.

1 Introduction

Simulating the flow of water on a terrain is a problem that has long been studied in geographic information science (GIS). It is common practice to derive drainage networks, channel lines, and catchment areas directly from a digital elevation model.

Naturally, these computations are affected by measurement errors of the elevations. A frequent way to deal with this imprecision is to model the elevation at a point of the terrain using stochastic methods [3, 7], leading to results that are not fully reliable. An approach taken in computational geometry [1, 2, 4] is to replace the exact elevation of each surface point by an imprecision interval, and computing the outcome of the most optimistic and pessimistic scenarios exactly. This is the approach we take in this paper.

When simulating water flow on terrain surfaces, it is assumed that water flows downward, in the direction of steepest descent. Most hydrological research in GIS uses a grid elevation model, in which each grid cell can drain to one or more of its eight neighbors, such as in the D-8 model [5]. For a discussion about the most common flow direction models see Tarboton [6]. When the surface is represented by a polyhedral terrain, the flow of water can be traced across the surface of a triangle, as discussed by Yu et al. [8].

Definitions. We define an ***imprecise terrain*** T as a possibly non-planar geometric graph in $\mathbb{R}^2$ in which each vertex $v \in \mathbb{R}^2$ has an imprecise third coordinate, which represents its ***elevation***. We denote the bounds of the elevation of v with $low(v)$ and $high(v)$. A ***realization*** R of an imprecise terrain T consists of the given graph together with an assignment of elevations to vertices such that for each vertex v its elevation $elev_R(v)$ is at least $low(v)$ and at most $high(v)$. We denote the set of all realizations of an imprecise terrain T with $\mathcal{R}_T$. We study the flow of water on imprecise terrains in two different models.

In the ***network model*** we assume that water only flows along the edges of the realization. The steepness of descent along an edge (p, q) in a realization R is defined as $\sigma_R(p, q) = (elev_R(p) - elev_R(q))/|pq|$. The water that arrives at a particular vertex p, flows to the neighbor q, such that $\sigma_R(p, q)$ is positive and maximal over all edges incident to p. For simplicity of exposition, we assume that this steepest descent neighbor is always unique and that edges are never horizontal in the realizations considered. If water flows from p to q in a realization R we write $p \xrightarrow[R]{} q$. If a vertex does not have a lower neighbor we call it a ***local minimum***.

The ***watershed*** of a vertex q in a realization R is defined as the set $\mathcal{W}(R, q) = \{p : p \xrightarrow[R]{} q\}$. The ***potential watershed*** of a vertex q in a terrain T is $\mathcal{W}_\cup(q) = \bigcup_{R \in \mathcal{R}_T} \mathcal{W}(R, q)$, that is, it is the set of points p for which there exists a realization R, such that water flows from p to q.

If the graph in the (x, y)-domain is a planar triangulation, then we can also consider the case that water flows across the polyhedral terrain represented by a realization. We call this the ***surface model***. The water that arrives at a particular point on this surface now flows in the true direction of the steepest descent, possibly across the interior of a triangle.

Results. In Section 2 we give a simple $O(n \log n)$ time algorithm to compute the potential watershed of a vertex in the network model, where n is the number of edges of the graph. The analysis also shows that for every vertex p, there is a realization of the terrain

*Utrecht University, The Netherlands, anne@cs.uu.nl. This work has been supported by the Netherlands Organisation for Scientific Research (NWO) under RIMGA.

†Dept. of Computer Sc., TU Eindhoven, the Netherlands

‡Computer Science Department, University of California, Irvine, USA, mloffler@uci.edu. Funded by the U.S. Office of Naval Research under grant N00014-08-1-1015.

§Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, rodrigo.silveira@upc.edu. Supported by NWO.

in which the watershed of p is its complete potential watershed, i.e., $\mathcal{W}_\cup(p)$ is realizable. In contrast, we show in Section 3 that in the surface model the problem of deciding whether water can possibly flow from a given point p to a given point q is NP-hard.

2 Computing watersheds in the network model

Canonical realization. We first show that the potential watershed of a vertex q is realizable.

Definition 1 *The* **watershed-overlay** *of a set of watersheds $\mathcal{W}(R_1, q_1), ..., \mathcal{W}(R_k, q_k)$ is the realization R^* such that for every vertex v, we have that $elev_{R^*}(v) = high(v)$ if $v \notin \bigcup \mathcal{W}(R_i, q_i)$ and $elev_{R^*}(v) = \min_{i:v\in\mathcal{W}(R_i,q_i)} elev_{R_i}(v)$ otherwise.*

Lemma 1 *Let R^* be the watershed-overlay of $\mathcal{W}(R_1, q), \ldots, \mathcal{W}(R_k, q)$, then it holds that $\mathcal{W}(R^*, q)$ contains $\mathcal{W}(R_i, q)$, for any $i \in \{1, \ldots, k\}$.*

Proof. Let u be a vertex of the terrain, which is contained in one of the given watersheds. Let R_i be a realization from $R_1, ..., R_k$ such that $elev_{R^*}(u) = elev_{R_i}(u)$. To prove the lemma, we show that u is contained in $\mathcal{W}(R^*, q)$ by induction on increasing elevation of u in R^*. The base case is that u is equal to q, and in this case the claim holds trivially.

Now, consider the vertex v which is reached from u by taking the steepest descent edge in R_i. Since $elev_{R^*}(v) \le elev_{R_i}(v) \le elev_{R_i}(u) = elev_{R^*}(u)$, it holds that v lies lower than u in R^*.

If v is still the steepest descent neighbor of u in R^*, then, by induction, $v \in \mathcal{W}(R^*, q)$ and therefore $u \in \mathcal{W}(R^*, q)$. Otherwise, there is a vertex $\widehat{v}$ such that $\sigma_{R^*}(u, \widehat{v}) > \sigma_{R^*}(u, v)$. There must be an R_j such that $\widehat{v} \in \mathcal{W}(R_j, q)$, since otherwise, by construction of the watershed-overlay, we have $elev_{R^*}(\widehat{v}) = high(\widehat{v}) \ge elev_{R_i}(\widehat{v})$ and thus, $\sigma_{R_i}(u, \widehat{v}) \ge \sigma_{R^*}(u, \widehat{v}) > \sigma_{R^*}(u, v) \ge \sigma_{R_i}(u, v)$ and v would not be the steepest descent neighbor of u in R_i. Therefore, by induction, also $\widehat{v} \in \mathcal{W}(R^*, q)$ and, again, $u \in \mathcal{W}(R^*, q)$. □

The above lemma implies that for any vertex q, the watershed-overlay R^* of all possible realizations in $\mathcal{R}_T$, realizes the potential watershed of q, i.e., $\mathcal{W}_\cup(q) = \mathcal{W}(R^*, q)$. Therefore, we call R^* the ***canonical realization*** of the potential watershed $\mathcal{W}_\cup(q)$.

Algorithm. Next, we describe how to compute the canonical realization of $\mathcal{W}_\cup(q)$ for a given vertex q. The idea of the algorithm is to compute the vertices of $\mathcal{W}_\cup(q)$ and their canonical elevations in increasing order of elevation, similar to the way in which Dijkstra's shortest path algorithm computes distances from the source. Refer to Algorithm 1 for the general outline.

The algorithm uses a subroutine Expand(q', z'), which returns for a vertex q' and an elevation $z' \in [low(q'), high(q')]$ the set of neighbors P of q', such that for each $p \in P$, there exists a realization R with $elev_R(q') \in [z', high(q')]$, such that $p \xrightarrow[R]{} q'$. In particular, it returns tuples of the form (p, z), where z is the minimum elevation of p over all such realizations R. We will now explain the preprocessing step that allows an efficient computation of Expand(q', z').

Algorithm 1 ComputePWS(q)

```
1: Enqueue (q, z) with key z = low(q)
2: while the Queue is not empty do
3:    (q', z') = DequeueMin()
4:    if q' is not already in the output set then
5:       Output q' and set elev_{R*}(q') = z'
6:       Enqueue each (p, z) ∈ Expand(q', z')
7:    end if
8: end while
```

We define the ***slope diagram*** of a vertex p as the set of points $\widehat{q}_i = (d_i, high(q_i))$, such that q_i is a neighbor of p and d_i is its distance to p in the (x, y)-projection. Let $q_1, q_2, ...,$ be the neighbors of p indexed such that $\widehat{q}_1, \widehat{q}_2, ...$ appear in counter-clockwise order along the boundary of the convex hull in the slope diagram, starting from the leftmost point and continuing to the lowest point (for simplicity of exposition we ignore the remaining neighbors). Let z_i be the value where the supporting line of the edge $\overline{\widehat{q}_i, \widehat{q}_{i+1}}$ intersects the vertical axis of the slope diagram, see Figure 1. We denote with $Z(p)$ the resulting decomposition of $[low(p), high(p)]$ into intervals that is given by the z_i and annotated by the corresponding points q_i. We can precompute $Z(p)$ in time $O(d \log d)$ and space $O(d)$, where d is the vertex degree of p. Since the sum of vertex degrees is $O(n)$, this takes $O(n \log d_{\max})$ time and $O(n)$ space overall, where $d_{\max}$ is the maximum vertex degree in the terrain.

Now, for a neighbor p of q', we can compute its elevation as it should be returned by Expand(q', z') by computing the lower tangent to the convex hull in the slope diagram, which passes through the point $\widehat{q'} = (d', z')$, where d' is the distance to p in the (x, y)-

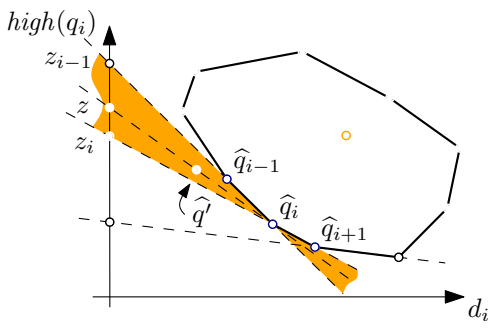


Figure 1: Slope diagram of the neighbors of p.

projection, see Figure 1. This can be done via a binary search on $Z(p)$ in time in $O(\log d_{\max})$. Intuitively, we annotated each interval of $Z(p)$, with the neighbor of p that the vertex q' has to compete with for being the steepest-descent neighbor if the elevation of p is in this interval. Doing this for all neighbors of q', we find that $\textsc{Expand}(q', z')$ runs in $O(d \log d_{\max})$ time, where d is the vertex degree of q'. We get the following lemma. We omit the full proof due to space limitations.

Lemma 2 *After precomputations in $O(n \log d_{\max})$ time and $O(n)$ space, $\textsc{Expand}(q, z)$ can be implemented to run in $O(d \log d_{\max})$ time.*

To prove the correctness of Algorithm 1 we use induction on the vertices extracted from the priority queue in the order of their extraction. The induction hypothesis says that for each extracted tuple (q', z'), it holds that there exists a realization R with $elev_R(q') = z'$ and $q' \xrightarrow[R]{} q$, such that this elevation is minimal. By Lemma 1, this implies that the algorithm outputs the canonical realization of $\mathcal{W}_\cup(q)$. As for the running time, it is easy to see that each vertex is expanded at most once. Using Lemma 2, we get the following theorem. We omit the full proof due to space limitations.

Theorem 3 *The algorithm $\textsc{ComputePWS}(q)$ computes the canonical realization of the potential watershed $\mathcal{W}_\cup(q)$ in time $O(n \log n)$, where n is the number of edges of the graph.*

3 NP-hardness in the surface model

In this section we sketch how to prove that deciding whether water potentially flows from a point s to another point t on an imprecise triangulated terrain, under the surface model, is NP-hard. The reduction is from 3-SAT; the input is an instance with n variables and m clauses. Globally, the construction consists of a grid with $O(m) \times O(n)$ squares, where each clause corresponds to a column and each variable to a row of the grid; the construction also contains some columns and rows that do not directly correspond to clauses and variables. The grid is placed across the slope of a "mountain" with shape similar to that of a pyramidal frustum. Figure 2 illustrates the construction. (The vertical faces in the illustration can easily be replaced by non-vertical triangulated slopes without affecting the construction.) A key element in the construction is the *divider* gadget (Figure 3, left), which is placed at every intersection of a clause column and a variable row. It consists of two imprecise vertices with a long edge between them and ensures that only if the two imprecise vertices are both at opposite extreme heights, can any water pass the divider gadget, otherwise it will flow to a local minimum. This way it carries over the (inverted) state of each imprecise vertex from left to right.

Across each divider gadget, water may flow in several courses, that may each veer off to the left or to the right, depending on the elevations of the imprecise vertices. We assume that the water takes the left courses if the variable is true, and the right courses if it is false. Now, to encode each clause, we let the water flow to a local minimum if and only if the clause is not satisfied. Figure 2 (right) shows an example.

In order to link the values of the imprecise vertices of the heights in the divider gadgets that belong to the same variable, we need to make sure that neighboring vertices have opposite extremal heights, just like in divider gadgets. For this, we use a *connector gadget*, which is basically the same construction as the divider gadget, see Figure 3 (right). As in the divider gadget, we only let the water escape if the heights of the imprecise vertices are at opposite extremes.

With this construction water can flow from s to t if and only if the 3-SAT formula can be satisfied.

4 Further Work

There are many related problems in the network model that cannot be discussed due to space limitations. We want to mention at least some of them.

Similar to the potential watershed of q, we can define the set of points that potentially *receive* water from q. Naturally, there is not always a canonical realization for this set, however, it can be computed in the same way as described in Section 2 using a priority queue that processes vertices in decreasing order of their maximal elevation, such that they would still receive water from q. Our definitions and algorithms naturally extend to sets of nodes. Therefore, we can also compute potential watersheds with respect to lakes or river beds.

Secondly, besides the potential flow paths, one may be interested in the question of whether water *always* flows between two vertices. We can define the set

$$\mathcal{W}_\cap(q) = \bigcap_{R \in \mathcal{R}_T} \mathcal{W}(R, q),$$

which is the set of points from which water flows to q in any realization. Observe that a vertex is contained in $\mathcal{W}_\cap(q)$ if and only if it does not have a potential flow path to a potential local minimum or a point outside $\mathcal{W}_\cup(q)$, which does not go through q. We can compute this set using the techniques described here. However, this definition may be very sensitive to flow paths being interrupted by overlapping imprecision intervals of neighboring vertices in relative flat terrain. Therefore, it is not clear if this is the right definition of persistent water flow.

Thirdly, note that for two given vertices p and q, such that $p \in \mathcal{W}_\cup(q)$, it is not necessarily true

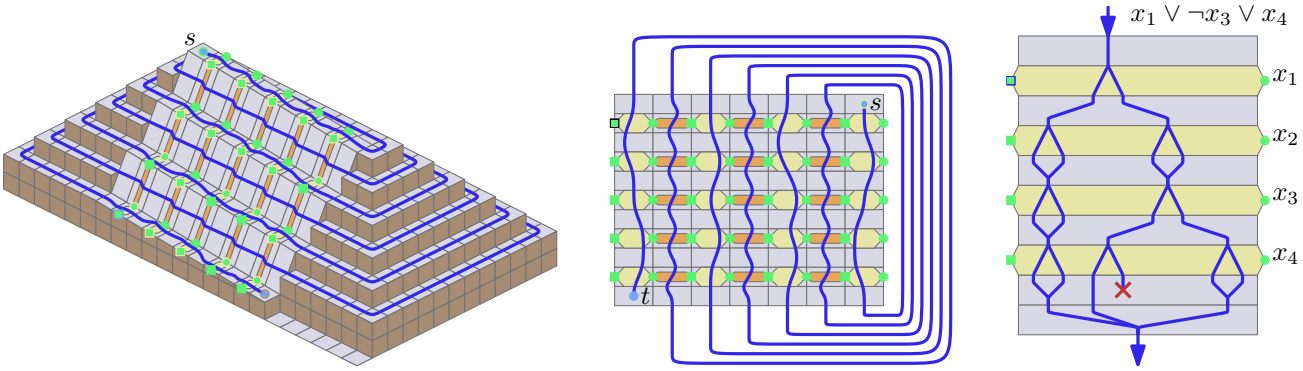


Figure 2: Left: Global view of the construction, showing the grid on the mountain slope. The fixed parts are shown in gray, the variable parts are shown yellow (for divider gadgets) and orange (for connector gadgets). Center: Top down view showing the locations of the gadgets, and the $n \times 2m$ green vertices, the only ones with imprecise heights. Right: Detail of a clause, which forms one of the columns of the grid in the center.

Figure 3: Left: A divider gadget consists of two imprecise vertices with an edge between them. Right: A connector gadget. The triangle needs to be much narrower, and the water streams need to be much closer to the center of the construction than in the picture.

that $\mathcal{W}_\cup(p) \subseteq \mathcal{W}_\cup(q)$. Therefore the potential watersheds of a terrain do not form a proper hierarchy. This makes it challenging to design a data structure that stores imprecise watersheds and answers queries about the flow of water between vertices efficiently.

Finally, the contrast between the results in Section 2 and Section 3 leaves room for further research questions, i.e., would it be possible to apply realistic input assumptions to make the potential flow computation in the surface model tractable? Another question is whether there exists a model of approximation for water flow and how it relates to the network model.

Acknowledgments. We are grateful to Chris Gray for many interesting and useful discussions.

References

[1] C. Gray and W. Evans. Optimistic shortest paths on uncertain terrains. In *Proc. 16th Canad. Conf. on Comput. Geom.*, pages 68–71, 2004.

[2] C. Gray, M. Löffler, and R. I. Silveira. Smoothing imprecise 1.5D terrains. In *Proc. 6th International Workshop on Approximation and Online Algorithms*, pages 214–226, 2009.

[3] F. Hebeler and R. Purves. The influence of elevation uncertainty on derivation of topographic indices. *Geomorphology*, 111(1-2):4 – 16, 2009.

[4] Y. Kholondyrev and W. Evans. Optimistic and pessimistic shortest paths on uncertain terrains. In *Proc. 19th Canad. Conf. on Comput. Geom.*, pages 197–200, 2007.

[5] J. O'Callaghan and D. Mark. The extraction of drainage networks from digital elevation data. *Computer vision, graphics, and image processing*, 28(3):323–344, 1984.

[6] D. Tarboton. A new method for the determination of flow directions and upslope areas in grid digital elevation models. *Water Resources Research*, 33(2):309–319, 1997.

[7] S. P. Wechsler. Uncertainties associated with digital elevation models for hydrologic applications: a review. *Hydrology and Earth System Sciences*, 11(4):1481–1500, 2007.

[8] S. Yu, M. van Kreveld, and J. Snoeyink. Drainage queries in TINs: from local to global and back again. In *Proc. 7th Int. Symp. on Spatial Data Handling*, pages 13A.1–13A.14, 1996.

The Number of k-Point Subsets Separable by Convex Pseudo-Circles

Dominique Schmitt* Jean-Claude Spehner*

Abstract

We show that any set S of n points in the plane contains $2kn-n-k^2+1-\sum_{i=1}^{k-1} a^{(i)}(S)$ subsets of k points that can be separated from the rest of S by convex pseudo-circles (where $a^{(i)}(S)$ denotes the number of i-sets of S). This value does not depend on the set of pseudo-circles.

1 Introduction

Given a finite set S of n points in the plane (no three of them being collinear) and an integer $k \in \{1, ..., n-1\}$, a classical geometric problem consists in searching subsets of k points of S that can be separated from the remaining points by different types of lines. Separation by a straight line has been extensively studied. In this case, the separable subsets are called k-sets. Dey [4] has shown that the number of k-sets of S (denoted by $a^{(k)}(S)$) is bounded by $O(nk^{\frac{1}{3}})$ and Tóth [10] has constructed sets with $n2^{\Omega(\sqrt{\log k})}$ k-sets. In fact, Dey's bound also holds when the separation lines form a family of pseudo-lines, that is x-monotone unbounded curves that pairwise intersect at most once. If the curves are x-monotone and if any two of them may have up to s intersection points (with s even), Buzaglo, Holzman, and Pinchasi [3] have shown that the number of separable k-point subsets is $O(n^{\frac{s}{2}}k^{\frac{s}{2}})$ and that the bound is tight.

When the separation lines are circles and no four points of S are cocircular, Lee [5] proved that S admits $2kn-n-k^2+1-\sum_{i=1}^{k-1} a^{(i)}(S)$ separable k-point subsets (those inside the circles). When the separation lines are pseudo-circles, *i.e.*, simple closed curves that pairwise intersect at most twice, and if all these pseudo-circles either pairwise intersect or enclose a same point of the plane, then the number of separable k-point subsets is $O(nk)$ [3]. The bound is tight.

In this paper we show that Lee's result holds true when the separation lines are convex pseudo-circles. This means that the number of subsets of k points of S that can be separated from the rest of S by convex pseudo-circles is an invariant of S: It does not depend on the set of pseudo-circles.

*Laboratoire LMIA, Université de Haute-Alsace, Mulhouse, France, {Dominique.Schmitt, Jean-Claude.Spehner}@uha.fr

2 Order-k Centroid Triangulations

A k-edge is a couple $(P, \{s, t\})$ of subsets of S such that $|P| = k$ and the straight line (st) separates P from $S \setminus (P \cup \{s, t\})$. Relations between k-edges and k-sets can be observed through the so-called k-set polygon of S (denoted by $g^k(S)$), which is the convex hull of the centroids of all k-point subsets of S. Indeed [1]:

Proposition 1 *(i) The centroid $g(T)$ of T is a vertex of $g^k(S)$ if and only if T is a k-set of S.*

(ii) The line segment $g(T)g(T')$ is an edge of $g^k(S)$ if and only if there exists a $(k-1)$-edge $(P, \{s, t\})$ of S such that $T = P \cup \{s\}$ and $T' = P \cup \{t\}$.

In the same way, circularly separable k-point subsets of S correspond to the regions of the order-k Voronoi diagram of S. The edges and vertices of this diagram are characterized by couples (P, Q) of subsets of S such that Q lies on a circle and P is inside the circle: For edges $|Q| = 2$ and $|P| = k-1$, and for vertices $|Q| = 3$ and either $|P| = k-1$ or $|P| = k-2$ (when no four points of S are cocircular) [5]. Taking the centroids of all circularly separable k-point subsets of S and connecting those corresponding to neighbor order-k Voronoi regions, we get a triangulation of the k-set polygon of S called the order-k (centroid) Delaunay triangulation of S [2, 8]. Its edges and triangles are characterized by the above defined couples (P, Q). Here we define the same kind of triangulation, but with convex pseudo-circles instead of circles.

Definition 1 *A couple (P, Q) of disjoint subsets of S is called a (convex) k-couple of S, if there exists a simple closed strictly convex (Jordan) curve γ such that Q lies on γ, P and $S \setminus (P \cup Q)$ lie respectively in the bounded and in the unbounded open region of the plane delimited by γ, and*

- *either $Q = \emptyset$ and $|P| = k$,*
- *or $|P| < k < |P \cup Q|$ and $2 \leq |Q| \leq 3$.*

The second item corresponds to three kinds of k-couples: $|Q| = 2$ and $|P| = k-1$, $|Q| = 3$ and $|P| = k-1$, $|Q| = 3$ and $|P| = k-2$.

The curve γ is said to define the k-couple (P, Q).

Definition 2 *Two distinct k-couples (P, Q) and (P', Q') of S are said to be compatible (with each other) if they admit two defining curves that intersect in at most two points (here a tangent point is considered as a double intersection point).*

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

Denoting, for every subset E of the plane, $\mathring{E}$ the relative interior of E and $conv(E)$ the convex hull of E, it is easy to see that:

Proposition 2 *(i) (P,Q) is a k-couple of S if and only if P and Q are two disjoint subsets of S such that $conv(P\cup Q)\cap S = P\cup Q$ and*

- *either $Q=\emptyset$ and $|P|=k$,*
- *or the points of Q are extremal points of $P\cup Q$, $|P|<k<|P\cup Q|$, and $2\leq|Q|\leq 3$.*

(ii) Two distinct k-couples (P,Q) and (P',Q') are compatible if and only if

$$\mathring{conv}((P\cup Q)\setminus P')\cap \mathring{conv}((P'\cup Q')\setminus P)=\emptyset.$$

Definition 3 *For every k-couple (P,Q) of S, let be the set of centroids of all k-point subsets of $P\cup Q$ that contain P. The convex hull of this set of centroids is called the k-set polygon of (P,Q) and is denoted by $g^k(P,Q)$.*

Let us describe the shape of the k-set polygon for every kind of k-couple (P,Q) of Definition 1:

- when $Q=\emptyset$, $g^k(P,Q)$ is the centroid of P,
- when $Q=\{s,t\}$, $g^k(P,Q)$ is the line segment $g(P\cup\{s\})g(P\cup\{t\})$,
- when $Q=\{r,s,t\}$ and $|P|=k-1$, $g^k(P,Q)$ is the triangle $g(P\cup\{r\})g(P\cup\{s\})g(P\cup\{t\})$; such a triangle is said of type 1,
- when $Q=\{r,s,t\}$ and $|P|=k-2$, $g^k(P,Q)$ is the triangle $g(P\cup\{r,s\})g(P\cup\{s,t\})g(P\cup\{t,r\})$; such a triangle is said of type 2.

When $k=1$, the 1-set polygons of the 1-couples of S are the points of S, the segments connecting the points of S, and the triangles with vertices in S and with no point of S inside. Notice that in this case all triangles are type-1. Furthermore, if the 1-couples are pairwise compatible, their corresponding points, open segments and open triangles are pairwise disjoint. The 1-set polygons of any maximal set of compatible 1-couples are then the set of vertices, edges, and faces of a triangulation of S. Conversely, every triangulation of S can be obtained in that way. We extend this to higher values of k.

Using basic centroid properties it is not hard to prove that:

Theorem 3 *If (P,Q) and (P',Q') are two distinct compatible k-couples, $\mathring{g}^k(P,Q)$ and $\mathring{g}^k(P',Q')$ are disjoint.*

The converse of Theorem 3 is generally wrong. It holds notably when $P=P'$ or when $P\cup Q=P'\cup Q'$.

Proposition 4 *The boundary of the k-set polygon of any k-couple (P,Q) with $|Q|\geq 2$ is composed of k-set polygons of k-couples that are compatible with (P,Q), compatible with each others, and compatible with every k-couple that is compatible with (P,Q).*

Observing precisely the boundary of each type of k-set polygon we can see that:

- the endpoints of a segment k-set polygon $g^k(P,\{s,t\})$ are the k-set polygons of the k-couples $(P\cup\{s\},\emptyset)$ and $(P\cup\{t\},\emptyset)$,
- the vertices of a type-1 triangle $g^k(P,\{r,s,t\})$ are the k-set polygons of the k-couples $(P\cup\{r\},\emptyset)$, $(P\cup\{s\},\emptyset)$, and $(P\cup\{t\},\emptyset)$. Its edges are the k-set polygons of the k-couples $(P,\{r,s\})$, $(P,\{s,t\})$, and $(P,\{t,r\})$,
- the vertices of a type-2 triangle $g^k(P,\{r,s,t\})$ are the k-set polygons of $(P\cup\{r,s\},\emptyset)$, $(P\cup\{s,t\},\emptyset)$, and $(P\cup\{t,r\},\emptyset)$. Its edges are the k-set polygons of $(P\cup\{r\},\{s,t\})$, $(P\cup\{s\},\{t,r\})$ and $(P\cup\{t\},\{r,s\})$.

Proposition 5 states the same kind of properties for the boundary of the k-set polygon $g^k(S)$ of the whole set S (they can be deduced from Proposition 1).

Proposition 5 *The edges and vertices of the k-set polygon of S are k-set polygons of k-couples that are compatible with each others and with every other k-couple of S.*

The next result is a kind of reciprocal of Proposition 4.

Proposition 6 *If $k\geq 2$ and if $(P_1,\{s_1,t_1\})$, $(P_2,\{s_2,t_2\})$, and $(P_3,\{s_3,t_3\})$ are distinct pairwise compatible k-couples whose k-set polygons form a triangle t then*

(i) t is the k-set polygon of a k-couple,

(ii) this k-couple is compatible with every k-couple that is compatible with $(P_1,\{s_1,t_1\})$, $(P_2,\{s_2,t_2\})$, and $(P_3,\{s_3,t_3\})$.

Proposition 6 is a strong property of k-set polygons of k-couples when $k\geq 2$. Transposed to the case where $k=1$, it would mean that a triangle with vertices in S cannot contain any point of S inside, which is obviously wrong.

Statement (ii) is a consequence of the following lemma:

Lemma 7 *Let $k\geq 2$ and let $(P,\{r,s,t\})$, $(P_1,\{s_1,t_1\})$, and $(P_2,\emptyset)$ be three compatible k-couples such that $g^k(P_1,\{s_1,t_1\})$ is an edge of $g^k(P,\{r,s,t\})$ and $g(P_2)$ is its opposite vertex.*

If $|P|=k-1$ (resp. $|P|=k-2$), every k-couple (P',Q') with $P'\neq P$ (resp. $P'\cup Q'\neq P\cup Q$) that is compatible with $(P_1,\{s_1,t_1\})$ and with $(P_2,\emptyset)$ is also compatible with $(P,\{r,s,t\})$.

This lemma is also the basis of the proof of the following fundamental result:

Theorem 8 *The k-set polygons of any maximal set of distinct compatible k-couples partition the k-set polygon of S in vertices, edges, and triangular faces.*

The triangulation obtained in this way is called an order-k centroid triangulation of S (see Figure 1).

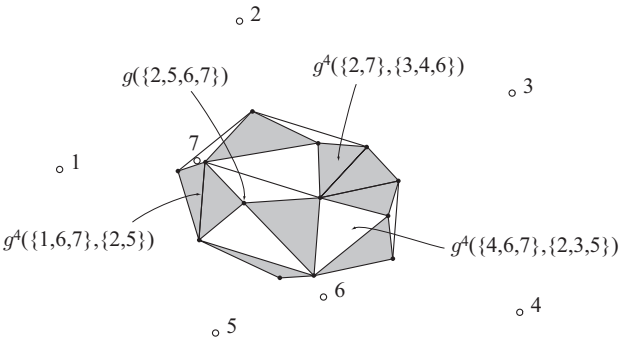


Figure 1: An order-4 centroid triangulation. Type-1 triangles are white and type-2 triangles grey.

An important consequence of Theorem 8, of Propositions 4 and 6, and of Lemma 7 is that:

Corollary 9 *If $\mathcal{C}$ is a maximal set of distinct compatible k-couples then the subset of all k-couples of $\mathcal{C}$ of the form (P, Q) with $Q = \emptyset$ (resp. $|Q| = 2$, resp. $|Q| = 3$ and $|P| = k-1$, resp. $|Q| = 3$ and $|P| = k-2$) is a maximal set of compatible k-couples of S of this form.*

3 Enumeration formulas

Let $\mathcal{T}^k$ be an order-k centroid triangulation of S ($k \geq 2$) and let τ be the set of k-couples (P, Q) whose k-set polygons are the type-2 triangles of $\mathcal{T}^k$ (*i.e.* $|Q| = 3$ and $|P| = k-2$). The k-couples of τ are also pairwise compatible $(k-1)$-couples. The $(k-1)$-set polygons of these $(k-1)$-couples are type-1 triangles and, from Theorem 8, they belong to a same order-$(k-1)$ centroid triangulation $\mathcal{T}^{k-1}$ of S. These triangles are the only type-1 triangles of $\mathcal{T}^{k-1}$: Otherwise there would exist a $(k-1)$-couple $(P', Q') \notin \tau$ with $|P'| = k-2$ and $|Q'| = 3$ compatible with the elements of τ. Since (P', Q') is also a k-couple compatible with the elements of τ, this would be in contradiction with Corollary 9. Hence:

Lemma 10 *The number of type-2 triangles of $\mathcal{T}^k$ is equal to the number of type-1 triangles of $\mathcal{T}^{k-1}$.*

There is also a relation between the vertices of $\mathcal{T}^k$ and the edges of $\mathcal{T}^{k-1}$.

Lemma 11 *$g(T)$ is a vertex of $\mathcal{T}^k$ if and only if there exists an edge $g^{k-1}(P, \{s, t\})$ in $\mathcal{T}^{k-1}$ with $P \cup \{s, t\} = T$.*

Proof. One can show that every vertex $g(T)$ of $\mathcal{T}^k$ is adjacent to at least one type-2 triangle. This triangle is then of the form $g^k(P, \{r, s, t\})$ with $T = P \cup \{s, t\}$. Thus, $g^{k-1}(P, \{r, s, t\})$ is a type-1 triangle of $\mathcal{T}^{k-1}$ and $g^{k-1}(P, \{s, t\})$ is one of its edges.

Conversely, if $g^{k-1}(P', \{s', t'\})$ is an edge of $\mathcal{T}^{k-1}$ with $(P', \{s', t'\}) \neq (P, \{s, t\})$, it is easy to see that $(P' \cup \{s', t'\}, \emptyset)$ is a k-couple compatible with $(P \cup \{s, t\}, \emptyset) = (T, \emptyset)$ and, from Corollary 9, $g(P' \cup \{s', t'\})$ is a vertex of $\mathcal{T}^k$. □

If, for a k-point subset T of S, the union of the edges and triangles of $\mathcal{T}^{k-1}$ of the form $g^k(P, Q)$ with $P \cup Q = T$ is not empty, then this union is called the domain of T in $\mathcal{T}^{k-1}$.

Clearly, every edge and every type-2 triangle of $\mathcal{T}^{k-1}$ belongs to one and only one domain. The edges of a type-2 triangle belong to the same domain as the triangle. The type-1 triangles belong to no domain. From Lemma 11, we then have:

Lemma 12 *The number of vertices of $\mathcal{T}^k$ is equal to the number of domains of $\mathcal{T}^{k-1}$.*

The right knowledge of the domains' shape will be needed to enumerate the vertices of $\mathcal{T}^k$.

Proposition 13 *(i) The vertices of $\mathcal{T}^{k-1}$ that belong to a same domain are the extremal points of this domain.*

(ii) Every domain is convex.

Proof. (i) Every vertex in the domain of T is of the form $g(T \setminus \{s\})$ since it is an endpoint of an edge of the form $g^{k-1}(T \setminus \{s, t\}, \{s, t\})$. Therefore, s is an extremal point of T and, by an homothety centered in $g(T)$, $g(T \setminus \{s\})$ is an extremal point of the domain.

(ii) We show that the line segment connecting any two vertices $g(T \setminus \{r\})$ and $g(T \setminus \{s\})$ of the domain of T also belongs to this domain. Let $g^{k-1}(T \setminus \{s, t\}, \{s, t\})$ be an edge of the domain of T with endpoint $g(T \setminus \{s\})$. If $t = r$ we are done. Otherwise, since $conv(T) \cap S = T$, since $|T| = k$, and since r, s, t are extremal points of T, $(T \setminus \{r, s, t\}, \{r, s, t\})$ is a $(k-1)$-couple of S. Now, $g^{k-1}(T \setminus \{s, t\}, \{s, t\})$ and $g(T \setminus \{r\})$ are respectively an edge and a vertex of the type-2 triangle $g^{k-1}(T \setminus \{r, s, t\}, \{r, s, t\})$. From Lemma 7, Proposition 4, and Theorem 3, only $(k-1)$-set polygons $g^k(P, Q)$ with $P \cup Q = T$ may cut the edge $g(T \setminus \{r\})g(T \setminus \{s\})$ of $g^{k-1}(T \setminus \{r, s, t\}, \{r, s, t\})$. Hence, this edge belongs to the domain of T. □

Theorem 14 *The number of vertices of any order-k centroid triangulation of S ($k \in \{1, \ldots, n-1\}$) is*

$$2kn - n - k^2 + 1 - \textstyle\sum_{i=1}^{k-1} a^{(i)}(S).$$

Proof. Consider a sequence $(\mathcal{T}^1, \ldots, \mathcal{T}^k)$ of order-1, ..., order-k centroid triangulations of S such that, for all $i \in \{2, \ldots, k\}$, the type-1 triangles of $\mathcal{T}^{i-1}$ correspond to the type-2 triangles of $\mathcal{T}^i$. Set $v^{(k)}, e^{(k)}, t_1^{(k)}$,

$t_2^{(k)}$ the respective numbers of vertices, edges, type-1 and type-2 triangles of $\mathcal{T}^k$. Since $\mathcal{T}^k$ is a triangulation of $g^k(S)$ and since the number of vertices of $g^k(S)$ equals the number $a^{(k)}(S)$ of k-sets of S, we get

$$e^{(k)} = 3v^{(k)} - a^{(k)}(S) - 3, \quad (1)$$

$$t_1^{(k)} + t_2^{(k)} = 2v^{(k)} - a^{(k)}(S) - 2. \quad (2)$$

Denoting by $e_T^{(k-1)}$ and $t_T^{(k-1)}$ the numbers of edges and of triangles of the domain of T in $\mathcal{T}^{k-1}$, Proposition 13 implies

$$e_T^{(k-1)} = 2t_T^{(k-1)} + 1.$$

Since every edge and every type-2 triangle of $\mathcal{T}^{k-1}$ belongs to one and only one domain, it then follows from Lemma 12,

$$e^{(k-1)} = 2t_2^{(k-1)} + v^{(k)}.$$

Thus, from Lemma 10,

$$e^{(k-1)} + e^{(k-2)} = 2(t_1^{(k-2)} + t_2^{(k-2)}) + v^{(k)} + v^{(k-1)}$$

and, using equations (1) and (2),

$$v^{(k)} - 2v^{(k-1)} + v^{(k-2)} = a^{(k-2)}(S) - a^{(k-1)}(S) - 2.$$

The result follows by induction on k and by the fact that $v^{(1)} = n$ (the vertices of $\mathcal{T}^1$ are the points of S) and that $v^{(2)} = 3n - a^{(1)}(S) - 3$ (the domains of $\mathcal{T}^1$ are the edges of $\mathcal{T}^1$). □

4 Discussion

Theorems 8 and 14 show that the number of subsets of k points of S that can be separated from the rest of S by convex pseudo-circles is equal to $2kn - n - k^2 + 1 - \sum_{i=1}^{k-1} a^{(i)}(S)$, and is an invariant of S. Furthermore, the couples (P, Q) such that P is a k-point subset separable by a convex pseudo-circle passing through Q correspond to the edges (when $|Q| = 2$) and to the type-1 triangles (when $|Q| = 3$) of an order-$(k+1)$ centroid triangulation. Their numbers can be deduced from proof of Theorem 14.

The convex pseudo-circles considered in this paper are closed curves. The results also hold for curves closing at infinity (as parabolas) if we consider the separable subset as the one belonging to the convex region of the plane delimited by the curve.

The separable subsets treated in this paper are closely related to the following ones: Let $\mathcal{F}$ be a family of subsets of S such that each member of $\mathcal{F}$ is the intersection of S with a convex set and, for any two members A and B of $\mathcal{F}$, $A \not\subseteq B$, $B \not\subseteq A$, and both $conv(A) \setminus conv(B)$ and $conv(B) \setminus conv(A)$ are connected (or empty). Pinchasi and Rote [7] proved that $\mathcal{F}$ contains at most $4\binom{n}{2} + 1$ members and that the bound is tight (within a constant multiplicative factor). Notice however that the constraints on the convex hulls in this definition are stronger than our compatibility condition.

In [5], Lee proposed an algorithm to construct the order-k Voronoi diagram by deducing it from the order-$(k-1)$ Voronoi diagram. This algorithm can be dualized to iteratively construct the order-k centroid Delaunay triangulation, starting with the classical (order-1) Delaunay triangulation [8]. In order to generate suitable bases for multivariate B-spline spaces, Liu and Snoeyink [6] proposed to extend this algorithm, starting with any (order-1) triangulation. When their algorithm succeeds in constructing a triangulation, then this triangulation is called a centroid triangulation. They proved that their algorithm actually works for $k \leq 3$ and conjectured that it works for all k. We can prove that our order-k centroid triangulations can be generated by such an algorithm and, conversely, that every execution of the algorithm generates such a triangulation [9]. This proves Liu and Snoeyink's conjecture.

References

[1] A. Andrzejak and E. Welzl. In between k-sets, j-facets, and i-faces: (i, j)-partitions. *Discrete Comput. Geom.*, 29:105–131, 2003.

[2] F. Aurenhammer and O. Schwarzkopf. A simple online randomized incremental algorithm for computing higher order Voronoi diagrams. *Internat. J. Comput. Geom. Appl.*, 2:363–381, 1992.

[3] S. Buzaglo, R. Holzman, and, R. Pinchasi. On s-intersecting curves and related problems. In *Symp. Comp. Geom.*, pages 79–84, 2008.

[4] T. K. Dey. Improved bounds on planar k-sets and related problems. *Discrete Comput. Geom.*, 19:373–382, 1998.

[5] D. T. Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Comput.*, C-31:478–487, 1982.

[6] Y. Liu and J. Snoeyink. Quadratic and cubic b-splines by generalizing higher-order Voronoi diagrams. In *Symp. Comp. Geom.*, pages 150–157, 2007.

[7] R. Pinchasi and G. Rote On the maximum size of an anti-chain of linearly separable sets and convex pseudo-discs. *Israel J. Math.*, 172:337–348, 2009.

[8] D. Schmitt and J.-C. Spehner. On Delaunay and Voronoi diagrams of order k in the plane. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 29–32, 1991.

[9] D. Schmitt and J.-C. Spehner. Generation of k-point subsets separable by convex pseudo-circles. In preparation.

[10] G. Tóth. Point sets with many k-sets. *Discrete Comput. Geom.*, 26(2):187–194, 2001.

Delaunay Topological Stability and Convex Hull of Point Sets with Coupled Uncertainties

Yonatan Myers and Leo Joskowicz *

Abstract

We address the topological stability and convex hull computation of a set of uncertain points with dependencies in the plane. Points are modeled with the Linear Parametric Geometric Uncertainty Model (LPGUM), a general and computationally efficient worst-case, first-order linear approximation of geometric uncertainty that supports dependence among uncertainties, described in previous papers. We define the Delaunay Topological Stability (DTS) of a set of n LPGUM points, modeled with k uncertainty parameters. We describe two efficient algorithms for testing DTS, which require $O(n \log n + nk)$ and $O(n \log n + nP_4(k))$ time for points with independent and dependent uncertainties, respectively, where $P_4(k)$ is the time for solving a quartic k-variable optimization. The convex hull of a set of DTS uncertain points, defined as the region enclosed by the boundaries of the union and intersection of all convex hull instances, is computed in $O(n \log r + nP_4(k) + rk^2 \log k)$, where r is the number of vertices of the convex hull.

1 Introduction

Geometric uncertainty is ubiquitous in many fields, such as mechanical engineering and wireless localization. Manufacturing, measurement, sensing and localization processes are intrinsically imprecise, and thereby introduce error and uncertainty. In contrast, their geometric models are usually exact and do not account for these inaccuracies. Modeling and computing geometric variability and its consequences is of scientific, technical, and economic importance.

Geometric uncertainty has been addressed in a variety of fields, from mechanical engineering [1, 3, 12] to computational geometry. A common approach to modeling geometric uncertainty is to use simple geometric entities, such as rectangles [7], circles [1, 3], or convex polygons [2] to bound point coordinate variations. Efficient algorithms for common problems, such as finding the smallest/largest enclosing circle/convex hull of independent uncertain points have been developed [8]. Recent papers study combinatorial properties using simple shapes to model point location uncertainties [4]. A key drawback of these models is that they cannot model mutually dependent uncertainties. Assuming error variations are independent often leads to an overestimation of the actual uncertainty [12].

We have recently developed the Linear Parametric Geometric Uncertainty Model (LPGUM) [6], a general, expressive, and computationally efficient worst-case, first-order, linear approximation of geometric uncertainty that handles **coupled uncertainties**. We have developed efficient algorithms for computing uncertainty zones of points and lines in the plane [6, 11], for relative points orientation, for point set distance problems, and for uncertain range queries [10].

In this paper we address the topological stability and convex hull computation of a set of uncertain points with dependencies in the plane. Both problems are of theoretical and practical interest in many fields. When point locations are uncertain, the relative order of points may change, thus giving rise to undesired point configuration changes.

We illustrate the importance of topological stability with a simple example. Fig. 1a shows a part defined by six vertices in their exact (nominal) locations. Fig. 1b shows the uncertainty zone of each vertex as a shaded rectangle, each representing the union of all possible vertex locations. Specific vertex instances give rise to a family of shapes. Fig. 1c shows an infeasible instance of the part, which results from a particular combination of vertex instances. Note that the instance is impossible, because the edges, v_1v_6 and v_2v_3, intersect: the orientation of the triangle, $\Delta(v_2, v_6, v_3)$, in this instance of the part is different from that in the nominal part, thus indicating a topological instability. The inconsistency results from the independent selection of extremal points of the vertices' uncertainty zones. Fig. 1d shows that when the point location uncertainties are dependent, e.g., with a global shearing error to the right, the inconsistency may not appear.

Topological stability of point sets has been widely studied [5]. Most papers define mappings of point sets or graphs between spaces of different dimensions that preserve properties such as connectivity, distances, and relative angles. We introduce Delaunay Topological Stability (DTS), a restricted and more practical notion of topological stability based on, point set nominal Delaunay triangulation. We show that DTS testing and convex hull computation of a set of DTS points can be performed efficiently in LPGUM.

*School of Engineering and Computer Science, The Hebrew University of Jerusalem, ISRAEL. Emails: yoni_m@cs.huji.ac.il, josko@cs.huji.ac.il

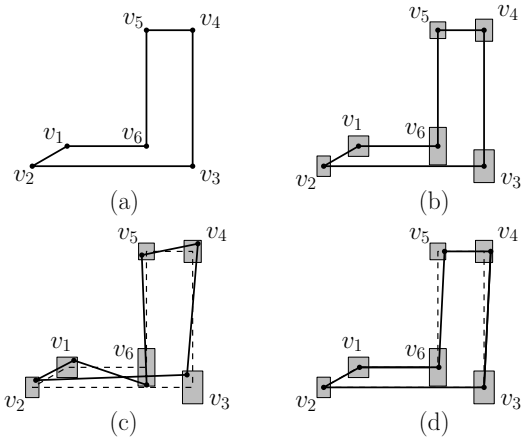


Figure 1: (a) Nominal polygonal part shape; (b) uncertainty envelopes of the vertices; (c) inconsistent part shape instance for independent point location uncertainties, and; (d) part shape instance for dependent point location uncertainties.

2 Geometric Uncertainty Model

To make the paper self-contained, we briefly introduce the basic LPGUM concepts. A parametric uncertainty model $(q, \bar{q}, \Delta)$ is defined as follows: Let $q = [q_1, q_2, \ldots, q_k]^T$ be a vector of k parameters over an *uncertainty domain* Δ. Each parameter $q_j \in \Re$ takes a value from a bounded *uncertainty interval* $\Delta_j = [q_j^-, q_j^+]$, where $q_j^- < q_j^+$ is associated with a nominal value $\bar{q}_j$. The parameters' uncertainty domain $\Delta = \Delta_1 \times \Delta_2 \ldots \times \Delta_k$ is the product of the parameters' uncertainty intervals – a hypercube in k-dimensional space. The *nominal parameter vector* $\bar{q} = (\bar{q}_1, \ldots, \bar{q}_k)$ is the parameter vector values with no uncertainty. Without loss of generality, we assume that the uncertainty intervals are zero-centered symmetric, i.e., $-q_j^- = q_j^+$ and $\bar{q}_j = 0$. Geometric objects with shared uncertainties are modeled by defining them over a common parametric uncertainty model $(q, \bar{q}, \Delta)$. The relation to each parameter determines the dependence between uncertainties. In mechanical engineering, k would be large for a complex part, and long chains of dimensions would lead to many points having many parameters in common.

An uncertain dimension $d(q)$ is defined by a nominal value $\bar{d}$ and a k-dimensional *uncertainty sensitivity vector* A_d over a parametric uncertainty model $(q, \bar{q}, \Delta)$. Entry $(A_d)_j$ is a constant that quantifies the sensitivity of the dimension to parameter q_j; it is zero when the dimension is independent of parameter q_j.

The LPGUM of dimension, $d(q)$, is $d(q) = \bar{d} + A_d q$. Its uncertainty zone is the interval, $[\min_{q\in\Delta} d(q), \max_{q\in\Delta} d(q)] \subset \Re$, whose endpoints are computed in $O(k)$ time [11]. Two dimensions, $d(q)$, $e(q)$ defined over a parametric uncertainty model $(q, \bar{q}, \Delta)$, are said to be *dependent*, iff there exists an index $0 < i \leq k$, for which $(A_d)_i \neq 0$ and $(A_e)_i \neq 0$, where A_d, A_e are their respective uncertainty sensitivity vectors, i.e.,when both dimension uncertainties depend on at least one common parameter.

An uncertain point, $v(q)$, is defined by a nominal location, $\bar{v}$, and a $2 \times k$ *uncertainty sensitivity matrix*, A_v, over a parametric uncertainty model, $(q, \bar{q}, \Delta)$. The uncertain point location is $v(q) = (v_x(q), v_y(q))^T$, where $v_x(q)$ and $v_y(q)$ are uncertain dimensions; the nominal point location is $\bar{v} = (\bar{v}_x, \bar{v}_y)^T$, where $\bar{v}_x$ and $\bar{v}_y$ are nominal values. Entry, $(A_v)_{i,j}$, is a constant that quantifies the sensitivity of coordinate, i, to parameter, q_j ($i = 1$ for x, $i = 2$ for y); it is zero if coordinate, i, is independent of parameter q_j. When the entire column, $(A_v)_j$, is zero, point $v(q)$ is independent of q_j.

The LPGUM of point, $v(q)$, is $v(q) = \bar{v} + A_v q$. Its uncertainty zone is the set of all locations for instances of parameter vector, q, $\mathcal{Z}(v) = \{v \mid v = \bar{v} + A_v q,\ q \in \Delta\}$. The point uncertainty zone envelope is a zonotope with at most $2k$ vertices computed in optimal $O(k \log k)$ time and $O(k)$ space [12]. Two LPGUM points are dependent when they both depend on at least one common parameter.

3 In-Circle Test

We define uncertain point set stability with respect to the nominal points Delaunay triangulation. In a Delaunay triangulation, the circles defined by the Delaunay triangles do not contain any other point of the set. We therefore first describe how to efficiently perform the in-circle test for LPGUM points.

Let $u(q), v(q), w(q)$ and $a(q)$ be four LPGUM points defined over a parametric uncertainty model, $(q, \bar{q}, \Delta)$, and let $C(q)$ be the circle defined by $u(q), v(q)$ and $w(q)$. The in-circle test determines whether there exists a parameter instance, $q^* \in \Delta$, such that $a(q^*)$ lies in the circle, $C(q^*) = (u(q^*), v(q^*), w(q^*))$.

When the LPGUM points are independent, the outer envelope of $C(q)$ is the union of three nominal circles, each tangential to the envelopes of the three LPGUM points [9]. Every circle has two points in its interior and one to its exterior. Thus, the in-circle test determines if the uncertainty envelope of $a(q)$ intersects the uncertainty envelope of $C(q)$, in $O(k)$ time, by testing for intersection of each of the point envelope vertices and edges versus the circle outer envelope.

For dependent LPGUM points, we use an algebraic approach. For nominal points, the in-circle test is solved by determining the sign of the determinant:

$$\begin{vmatrix} \bar{u}_x & \bar{u}_y & \bar{u}_x^2 + \bar{u}_y^2 & 1 \\ \bar{v}_x & \bar{v}_y & \bar{v}_x^2 + \bar{v}_y^2 & 1 \\ \bar{w}_x & \bar{w}_y & \bar{w}_x^2 + \bar{w}_y^2 & 1 \\ \bar{a}_x & \bar{a}_y & \bar{a}_x^2 + \bar{a}_y^2 & 1 \end{vmatrix}$$

where $\bar{u}, \bar{v}$ and $\bar{w}$ are in counter-clockwise order. A negative value indicates that $\bar{a}$ is outside the circle. Replacing the nominal points by LPGUM points, subject to the parameter's interval constraints, $q \in \Delta$, produces a quartic (fourth-degree) k-variable optimization problem, which can be solved in $P_4(k)$ time.

4 Point Set Delaunay Topological Stability

Let $S = \{s_1(q), \ldots, s_n(q)\}$ be a set of n LPGUM points in the plane, defined over a parametric uncertainty model, $(q, \bar{q}, \Delta)$. Let $\bar{S}$ be the corresponding nominal points set, and S_q the set of point instances of a particular parameter's instance, q. Let $\mathcal{DT}(\cdot)$ denote the Delaunay triangulation.

Definition 1 *The set S of LPGUM points is* Delaunay Topologically Stable *(DTS) iff, for every triangle, $\triangle(\bar{s}_a, \bar{s}_b, \bar{s}_c) \in \mathcal{DT}(\bar{S})$, there is a corresponding triangle, $\triangle(s_a(q), s_b(q), s_c(q)) \in \mathcal{DT}(S_q)$ for all $q \in \Delta$.*

We now show how the nominal Delaunay triangulation and the in-circle tests are used to determine whether a point set is DTS.

Theorem 1 *Let S be a set of n LPGUM points. Let $T_{abc} = \triangle(\bar{s}_a, \bar{s}_b, \bar{s}_c)$ be a triangle in $\mathcal{DT}(\bar{S})$. Let $\bar{s}_d$ be a neighbor of T_{abc}, such that the triangle $T_{adb} = \triangle(\bar{s}_a, \bar{s}_d, \bar{s}_b)$ is a triangle of $\mathcal{DT}(\bar{S})$. The set S is DTS, iff, $s_d(q)$ lies outside the circle defined by $s_a(q)$, $s_b(q)$ and $s_c(q)$, for all $q \in \Delta$.*

Proof. $\Rightarrow$: By contradiction, suppose that S is DTS but there exists a $q^* \in \Delta$ such that $d(q)$ lies in $C(q)$. Thus the triangle $\triangle(s_a(q^*), s_b(q^*), s_c(q^*)) \notin \mathcal{DT}(S_{q^*})$. This means that S is not DTS.
$\Leftarrow$: By contradiction, suppose that $d(q)$ does not lie in $C(q)$ for any $q \in \Delta$ but that S is not DTS. The latter implies one of two possibilities: The first is that there exists a parameter instance, $q^* \in \Delta$, such that connecting the points of S_{q^*} in the same order as in $\mathcal{DT}(\bar{S})$ yields a legal triangulation that is not $\mathcal{DT}(S_{q^*})$. But this implies that $d(q^*)$ lies in the circle $C(q^*)$ – a contradiction. The second possibility is that there exists a parameter instance, q^{**}, such that connecting the points of $S_{q^{**}}$ in the same order as in $\mathcal{DT}(\bar{S})$ does not yield a legal triangulation. Since the LPGUM point locations are defined by continuous functions, there must be a parameter instance, $q^* \in \Delta$, such that connecting the points of S_{q^*} in the same order as in $\mathcal{DT}(\bar{S})$ yields a legal triangulation, which amounts to the first scenario.

We conclude that, if $s_d(q)$ is outside $C(q)$ for all $q \in \Delta$, then connecting the vertices of S_q in the same order as in $\mathcal{DT}(\bar{S})$ yields the Delaunay triangulation $\mathcal{DT}(S_q)$. Thus, S is DTS. □

Definition 2 *The set of LPGUM points, S, is* Strongly Delaunay Topologically Stable *(SDTS) iff, for every triangle $\triangle(\bar{s}_a, \bar{s}_b, \bar{s}_c) \in \mathcal{DT}(\bar{S})$, and every one of its neighbors, $\bar{s}_d$, $s_d(q)$, lies outside the circle, $C(q) = (s_a(q), s_b(q), s_c(q))$, when all four points are considered to be independent.*

In SDTS, the dependencies between the point uncertainties are not taken into account. Thus, by definition, SDTS is stronger and implies DTS, since the dependencies between the point location uncertainties restrict uncertainty zone extremal point selection.

In many cases, requiring DTS of the entire point set S is too strong, as topological stability may only be necessary with respect to a subset, S', of the points, e.g., those in the outer contour of a part, or on the convex hull of a region. The subset, S', can be interpreted as the anchor or stable subset of S.

Definition 3 *The set, S, of LPGUM points is Partially Delaunay Topologically Stable (PDTS) with respect to a subset of S' iff, S' is DTS and $\forall\ s_i(q) \in S \backslash S'$, the set $S' \cup \{s_i(q)\}$ is DTS.*

5 Delaunay Topological Stability Testing

Theorem 1 directly implies that a linear number of in-circle tests suffice to determine if a set is DTS, as a Delaunay triangulation is a planar subdivision whose complexity is linear in the number of points.

To determine if S is SDTS, we first compute $\mathcal{DT}(\bar{S})$. Then, for every triplet $(s_a(q), s_b(q), s_c(q))$ of triangle $\triangle(\bar{s}_a, \bar{s}_b, \bar{s}_c) \in \mathcal{DT}(\bar{S})$ and for every point $s_d(q)$ for which $\bar{s}_d$ is a neighbor of $\triangle(\bar{s}_a, \bar{s}_b, \bar{s}_c)$, we do an independent in-circle test, i.e., we check whether $s_d(q)$ is outside the uncertain circle, $C(q) = (s_a(q), s_b(q), s_c(q))$, when the points are considered to be independent.

Theorem 2 *Let S be a set of n LPGUM points. Determining if S is SDTS requires $O(n \log n + nk)$ time.*

To test whether S is DTS, we first check whether it is SDTS with independent points in-circle tests whose complexity is $O(k)$. If S is not SDTS, we record the $m \leq n$ independent in-circle tests failed, and perform the dependent in-circle tests on them, each in $P_4(k)$ time.

Theorem 3 *Let S be a set of LPGUM points. Determining if S is DTS requires $O(n \log n + nk + mP_4(k))$ time; m is the number of failed independent point, in-circle tests, and $P_4(k)$ is the time required to solve a single dependent point in-circle test.*

From this theorem, we derive a method to compute the Delaunay triangulation of a DTS set of uncertain points, S. First, we compute the nominal Delaunay triangulation $\mathcal{DT}(\bar{S})$. Then, for every $\overline{\bar{s}_i \bar{s}_j} \in \mathcal{DT}(\bar{S})$, we connect $s_i(q)$ and $s_j(q)$ with the LPGUM segment $\lambda s_i(q) + (1 - \lambda) s_j(q)$, $\lambda \in [0, 1]$, whose uncertainty envelope is computed in $O(k^2 \log k)$ time [11]. Since there are $O(n)$ edges, this takes $O(nk^2 \log k)$ time, in addition to the time for testing that S is DTS.

Theorem 4 *Let S be a set of LPGUM points in the plane, and let S' be a subset of S of size r. Testing if S is Partially Delaunay Topologically Stable with respect to S' takes, on average, a time of $O(n \log r + nP_4(k))$, and $O(n \log r + (n - r)rP_4(k))$, in the worst case.*

To test if S is PDTS with respect to S', we test if S' is DTS, and then for every $s_i(q) \in S \backslash S'$, if $S' \cup \{s_i(q)\}$ is DTS, by adding $\bar{s}_i$ to $\mathcal{DT}(\bar{S}')$, and performing in-circle tests for its neighboring triangles. The average cardinality of Delaunay triangulation vertices is 6, which leads to the stated run time.

6 Convex Hull of DTS Uncertain Points

We now consider the problem of computing the convex hull of a set of uncertain points with dependencies. The problem has been investigated for independent uncertain points [7], but not for points with dependent uncertainties. Note that, without restrictions, there can be exponentially many topologically different convex hulls, for a set of uncertain points, S, as all points subsets of S can form the convex hull, depending on each point uncertainty. Consequently, it is of practical interest to define the convex hull of uncertain points that are topologically stable, with respect to the convex hull of the nominal points, $\mathcal{CH}(\bar{S})$.

The convex hull of a set of uncertain points is defined as the region enclosed by the boundaries of the union and intersection of all convex hull instances. Points outside (inside) the outer (inner) convex hull boundary are always outside (inside) any convex hull instance of points in S. Other points are uncertain.

Definition 4 *The union and intersection convex hulls of a set S of n LPGUM points are:*

$$\mathcal{UCH}(S) = \bigcup_{q\in\Delta} \mathcal{CH}(S_q) \qquad \mathcal{ICH}(S) = \bigcap_{q\in\Delta} \mathcal{CH}(S_q)$$

The union convex hull is a tight boundary on the set of all points that lie in $\mathcal{CH}(S_q)$ for $q \in \Delta$. The intersection convex hull is the set of all points that are in $\mathcal{CH}(S_q)$ for any $q \in \Delta$.

We compute the union/intersection convex hull of a set S of DTS LPGUM points as follows. We compute the nominal convex hull $H = \mathcal{CH}(\bar{S})$ and its vertices, $S_H \subseteq S$ in $O(n \log r)$ time. Then, we test if S is PDTS with respect to S_H. If so, we connect the points of S_H with LPGUM segments in their order of appearance in H. We construct each segment uncertainty envelope in $O(k^2 \log k)$ time [11]. The union (intersection) convex hull is the outer (inner) boundary.

Theorem 5 *The union/intersection convex hull of a set, S, of n dependent LPGUM points takes, on average, $O(n \log r + nP_4(k) + rk^2 \log k)$ time and $O(n \log r + (n-r)rP_4(k) + rk^2 \log k)$, in the worst case, where r is the number of points on $\mathcal{CH}(\bar{S})$.*

7 Conclusion

Determining the topological stability of a set of points whose locations are uncertain and interdependent is of theoretical interest, and has applications in many fields. Most existing point location uncertainty and point set topological stability models are either too restrictive or are computationally expensive.

We have introduced the Delaunay Topological Stability (DTS) and its variants, Partial and Strong DTS, for points in the Linear Parametric Geometric Uncertainty Model (LPGUM). We described efficient algorithms for testing if a set of LPGUM points is DTS, and for computing the uncertain Delaunay triangulation and uncertain convex hull of DTS point sets.

A key property of a DTS set of LPGUM points is that the point set retains its inner structure and relative points positions (or part of it) despite the interdependent point location uncertainties. This is a realistic assumption for many applications, as it does not restrict the magnitude of nominal values and their variations, and supports efficient computation. It may prove useful for other hard/unsolved problems, e.g., the minimum diameter and the maximum closest uncertain point set problems [10]. Future work includes computing the convex hull of non-PDTS point sets, and the uncertain Voronoi diagram of DTS point sets.

References

[1] S. Akella and M. Mason. Orienting toleranced polygonal parts. *Int. J. of Robotics Research* 19(12):1147–1170, 2000.

[2] R.C. Brost and R.R. Peters. Automatic design of 3D fixtures and assembly pallets. *Proc. IEEE Int. Conf. Robotics and Automation*, USA 1996.

[3] J. Chen, K. Y. Goldberg, et al. Computing tolerance parameters for fixturing and feeding. *The Assembly Automation Journal* 22:163–172, 2002.

[4] M. de Berg, E. Mumford, and M. Roeloffzen. Finding structures on imprecise points. *Proc. 26th European Workshop on Computational Geometry*, pp 85–88, Dortmund, Germany, 2010.

[5] A. du Plessis, T. Wall. *The Geometry of Topological Stability*. London Math. Society Monographs New Series. Oxford Univ. Press, 1996.

[6] L. Joskowicz, Y. Ostrovsky-Berman, and Y. Myers. Efficient representation and computation of geometric uncertainty: The linear parametric model. *Precision Engineering* 34(1):2–6, 2010.

[7] M. Löffler and M. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica* 56(2):235–269, 2010.

[8] M. Löffler and M. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Computational Geometry: Theory and Applications* 43(4):419–433, 2010.

[9] Y. Myers and L. Joskowicz. Circles with independent and dependent uncertainties. *Proc. 26th European Workshop on Computational Geometry*, pp 229–232, Dormuth, Germany, 2010.

[10] Y. Myers and L. Joskowicz. Point distance and orthogonal range problems with dependent geometric uncertainties. *Proc. 14th ACM Symp. on Solid and Physical Modeling*, NY, USA, 2010.

[11] Y. Myers and L. Joskowicz. Uncertain geometry with dependencies. *Proc. 14th ACM Symp. on Solid and Physical Modeling*, NY, USA, 2010.

[12] Y. Ostrovsky-Berman and L. Joskowicz. Tolerance envelopes of planar mechanical parts with parametric tolerances. *Computer-Aided Design* 37(5):531–544, 2005.

Cannibal Animal Games: a new variant of Tic-Tac-Toe

Jean Cardinal Sébastien Collette Hiro Ito† Matias Korman Stefan Langerman
Hikaru Sakaidani† Perouz Taslakian

Abstract

This paper presents a new partial two-player game, called the *cannibal animal game*, which is a variant of Tic-Tac-Toe. The game is played on the in nite grid, where in each round a player chooses and occupies free cells. The rst player Alice, who can occupy a cell in each turn, wins if she occupies a set of cells, the union of a subset of which is a translated or rotated copy of a previously agreed upon polyomino P (called an *animal*). The objective of the second player Bob is to prevent Alice from creating her animal by occupying in each round a translated or rotated copy of P. An animal is a *cannibal* if Bob has a winning strategy, and a *non-cannibal* otherwise. This paper presents some new tools, such as the *bounding strategy* and the *punching lemma*, to classify animals into cannibals or non-cannibals. It is also shown that the *pairing strategy* also works for this problem.

1 Introduction

Studying variants of the Tic-Tac-Toe game are interesting problems in the area of recreational mathematics [1, 2, 3, 5, 6, 7, 8, 9]. Probably the most studied among these games is an achievement game, a somewhat generalized Tic-Tac-Toe, presented by Harary [3, 5]. A *polyomino* or an *animal* is a set of connected cells (sharing an edge) of the in nite grid. In such games, two players Alice and Bob alternatively occupy one cell of the in nite grid in each round of the game, and the player who is the rst to occupy a translated copy of the given animal is a winner (we will always assume that Alice is the rst player). In these games Bob cannot win, hence his objective is to obstruct Alice's achievement.

Here we present a new achievement game called the *cannibal animal game*. As with Harary's achievement game, it is played on the in nite grid whereby players alternate turns to occupy free cells of the grid. This means that in each round the player must choose grid cells that are not yet occupied; hence, occupied regions do not intersect. Moreover, once a cell is occupied, it remains so until the end of the game. In contrast to the generalized Tic-Tac-Toe, the cannibal animal game is a *partial game*: the roles and legal moves of Alice and Bob are di erent. Alice's legal move is to occupy one cell of the in nite grid in each round, and she wins if she occupies a translated copy of an animal given beforehand (this move is the same as that of the rst player of Harary's generalized Tic-Tac-Toe). Bob's role and allowed moves, however, are di erent: in each round he must occupy a copy of the given animal (hence occupy a subset of the grid cells), and his objective is to prevent Alice from achieving the animal. The animal achieved or that Bob occupies may be a translation, a mirror image and/or a 90, 180, or 270-degree rotation of the given animal. Each such translation/rotation is called a *copy* of the animal and n-cell-animal is an animal consisting of n cells. Figure 1 shows an example of the progress of the game where the animal is *El*, an L-shaped triomino.

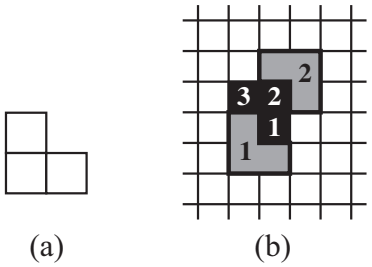


Figure 1: (a) The animal El (an L-shaped triomino), (b) An example of the progress of the game: cells depicted in black are occupied by Alice, and animals depicted in gray correspond to Bob's moves. In both cases, the numbers on the cells represent the order in which the cells are occupied. In the example, Alice wins.

We call an animal a *cannibal* or a *loser* if Bob has a winning strategy (Bob's animal eats Alice's animal) and a *non-cannibal* or a *winner* otherwise. And hence the game is called the *cannibal animal game*.

Our Results. In this paper we study the following animals (see Figure 2 for examples): $R(n, m)$ is an

Department of Computer Science, Université Libre de Bruxelles (ULB), Belgium, {jcardin, sebastien.collette, mkormanc, stefan.langerman, perouz.taslakian}@ulb.ac.be

†School of Informatics, Kyoto University, Japan, {itohiro@, sakaidan@lab2.}kuis.kyoto-u.ac.jp

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

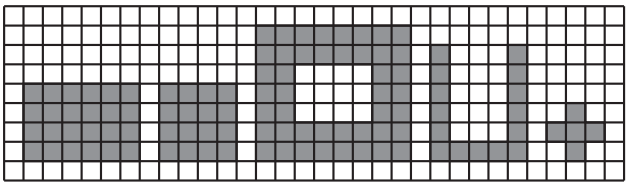


Figure 2: Examples of animals: $R(4,6)$, $S(4)$, $O(7,8,2)$, $U(6,5,1)$, and $X(3)$ (from left to right).

nm-cell-animal of an n m rectangle. $R(n,n)$ is sometimes expressed as $S(n)$ (or an n n square). $O(n,m,k)$ (for $k < \min\{n/2, m/2\}$) is a $2k(n+m$ $2k)$-cell-animal having the shape of $R(n,m)$ but with a hole of $(n$ $2k)$ $(m$ $2k)$ rectangle in the center (that is, an O-shaped polyomino whose thickness is k). $U(h,w,k)$ (for $k < \min\{h, w/2\}$) is a $k(2h+w+2k)$-cell-animal having a U-shape with height h, width w, and thickness k. $X(n)$ (or n-cross) is a $(2n$ $1)$-cell-animal consisting of vertical and horizontal cells, each of length n, that cross each other at the center cells.

1. The following animals are cannibals:
   (a) $S(n)$ with holes if at least one of the holes is at least $\lfloor n/4 \rfloor$ cells away from the boundary for $n \geq 4$ (and no hole is on the boundary),
   (b) $O(n,m,k)$ for $n,m,k \in \mathbb{N}$,
   (c) $U(h,w,1)$ for $h,w \in \mathbb{N}$, except $U(2,4,1)$.
2. The following animals are non-cannibals:
   (a) Animals with at most three cells,
   (b) $R(n,m)$ for any $n,m \in \mathbb{N}$,
   (c) $X(3)$.

2 Cannibal animals (losers) and pairing strategy

In this section we demonstrate a strategy for Bob that prevents Alice winning for a few families of animals. To do this, we will use the idea of *pairing strategy* that is used in many other combinatorial games. In what follows we will show that this approach also works for our cannibal animal game. We start with a simple strategy for Bob that works for the $O(n,m,k)$ animal:

Theorem 1 *$O(n,m,k)$ is a cannibal for any $n,m \geq 3$ and $k < \min\{n/2, m/2\}$.*

Proof. Bob virtually partitions the playing-board into blocks of size $(n+k)$ $(m+k)$. That is, we de ne the block B_{ij} as the rectangle $[i(n+k), (i+1)(n+k)$ $1]$ $[j(m+k), (j+1)(m+k)$ $1]$ (as shown in Figure 3). The strategy for Bob is to place his animal inside the block where Alice played her last move. After Alice plays, Bob checks which block her last move belongs to; if he has already played an animal in that same block, he simply plays in an arbitrary empty block (e.g., Bob's 4th move in Figure 3). Note that since the playing board is in nite, Bob can always play these moves. With this strategy, Alice clearly cannot construct a copy of $O(n,m,k)$.

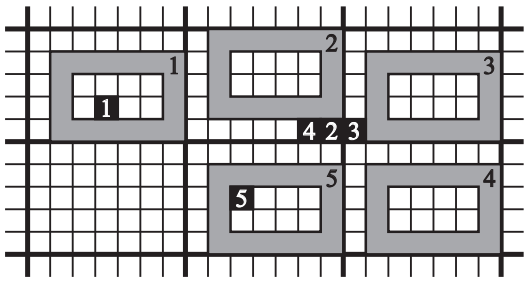


Figure 3: Winning strategy for Bob for $O(n,m,k)$ (in this example, for $O(4,6,1)$). Alice's moves are marked in black and Bob's in gray. The numbers on the cells represent the order in which the cells are occupied. Since the block inside which Alice's 4th move is played already includes Bob's animal, Bob's 4th move is played in another arbitrary block.

This strategy is also useful for other animals. Recall that by Theorem 6 squares are non-cannibals. Surprisingly, the removal of a single interior cell from a square animal can transform it into a cannibal.

Lemma 2 *For any integer $n \geq 4$, let A be an n n square animal in which a single cell whose distance to the boundary is at least $\lfloor n/4 \rfloor$ units has been removed. Then A is a cannibal.*

Proof. The proof is analogous to the proof of theorem 1. This time we partition the board into blocks of size $(n+\lfloor (n$ $1)/2 \rfloor)$ $(n+\lfloor (n$ $1)/2 \rfloor)$. If the hole (removed cell) is at least $\lfloor n/4 \rfloor$ units away from the boundary, then Bob can always play his animal inside the same block as Alice's last move (details omitted in this version).

Assume that Alice is able to construct a copy of the animal on the board. By the porthole principle, there would be a block in which Alice's pieces form a square of size at least $\lceil n/2 \rceil$ $\lceil n/2 \rceil$ (possibly with one cell removed). However, this cannot occur since Bob also occupies the same block with an n n square.

In some cases, we might also need a more careful partitioning of the grid into blocks:

Theorem 3 *For any $h,w \in \mathbb{N}$ (other than $(h,w) = (2,4)$), the $U(h,w,1)$ animal is cannibal.*

Proof. In this case, Bob virtually partitions the playing board into blocks of size $(w+k)$ h. But if he arranges these blocks naively, there might be "cracks" between Bob's animals in which Alice could construct

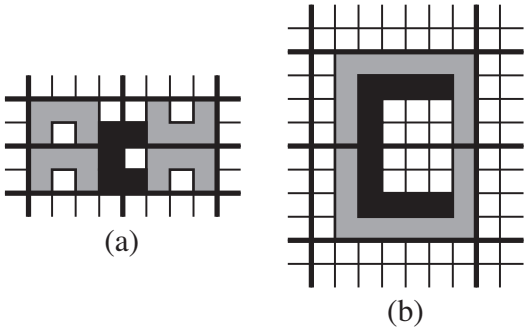


Figure 4: Examples of failed partitions.

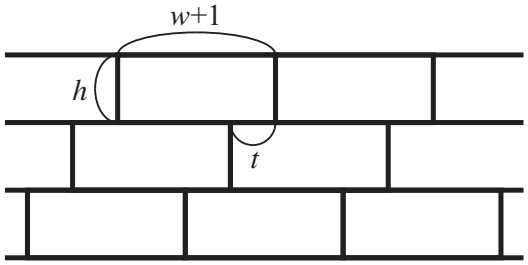


Figure 5: Tiling and shift size t.

her animal (see Figure 4). To avoid such cracks, Bob must slant his partition, thus tiling the grid with blocks with a shift of size (distance) t (Figure 5). We de ne the block $B_{i,j}$ as the rectangle $[i(w+1)+jt, i(w+1)+w+jt] \quad [jh, jh+h \quad 1]$. The exact value of the slant depends on the parameters w and h:

$h = 2$ **(and** $w \neq 4$**):** $t = 2$.

$h \geq 3$ **and** $2h \quad 2 \geq w \geq h \quad 2$**:** $t = \lfloor (w+1)/2 \rfloor$.

Otherwise: No slant is necessary (i.e., $t = 0$).

It is easy to show that with such a partition, Alice will be unable to construct her animal (details omitted in this version).

We now introduce another idea to generate new cannibal animals from known cannibal animals. Let A be an animal and let C be a subset of cells of A. Then $A \setminus C$ is an animal created by removing C from A. We say that C is an *outer piece* if we can locate a second copy of $A \setminus C$ that covers a part of the removed piece C of the rst copy; we call C an *inner piece* otherwise. See Figure 6.

Notice that even if C and C' are both inner pieces, $C \cup C'$ may be outer. If C is an outer piece, then for any superset C' of C is also outer.

Lemma 4 (Punching Lemma) *Let A be a cannibal and let C be an inner piece of A. The animal $A \setminus C$ is also a cannibal.*

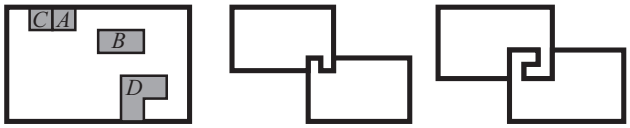


Figure 6: A and B are inner pieces. C and D are outer pieces since a second copy covers a part of the piece as seen in the right examples.

Proof. Assume otherwise that $A \setminus C$ is a non-cannibal; then Alice would have a winning strategy, i.e., she will be able to construct a copy of $A \setminus C$ without Bob preventing it. Consider now the removed piece C of the animal Alice constructed. Because C is inner, this position cannot be occupied by Bob. Moreover, Alice can occupy this position in another round to form animal A. Thus, a contradiction.

Note that the reciprocal is not always true (see for example Theorem 6 and Lemma 2). As a simple application of this lemma, we have the following result:

Theorem 5 *For any integer $n \geq 4$, let S' be an animal $S(n)$ in which any number of interior cells have been removed. If at least one of the removed cells has distance $\lfloor n/4 \rfloor$ or more to the boundary, then S' is a cannibal.*

3 Non-cannibal animals (winners) and the bounding strategy

In this section we give a few families of non-cannibal animals. We rst introduce a concept on common intersection:

Definition 1 *An animal P is called* 2-Helly *[4] if for any family $\mathcal{A}$ of copies of P such that $A \cap A' \neq \emptyset$ for any $A, A' \in \mathcal{A}$, the intersection $\bigcap_{A \in \mathcal{A}} A$ is nonempty.*

It is easy to see that $R(n, m)$ for any n and m is 2-Helly, while none of the other animals that we study are.

Theorem 6 *Any 2-Helly animal is a non-cannibal.*

For proving this theorem, we rst prove a restricted version as follows:

Lemma 7 *In any nite board, any 2-Helly animal P is a non-cannibal provided that at least one copy P can be occupied on the board.*

Proof. At the beginning of each round we de ne $\mathcal{S} = \{s_1, \ldots, s_k\}$ as the set of copies of P not occupied by Bob that t in the board (note that some of these positions may be occupied by Alice's previous moves). The set $\mathcal{S}$ will be treated as a set of potential positions in which Alice may form her animal. Note that Bob's

moves must be at some $s \in \mathcal{S}$. Also, let $\mathcal{S}' \subseteq \mathcal{S}$ be the set of animals that stab all elements of $\mathcal{S}$ (that is, $s' \in \mathcal{S}' \Leftrightarrow s' \cap s \neq \emptyset$, $\forall s \in \mathcal{S}$).

Note that the set $\mathcal{S}$ initially is nonempty at the beginning of the game, and whenever Bob plays, the size of S is reduced. However, notice that S will only become empty if Bob can place his copy occupying the cells of some $s' \in \mathcal{S}'$. Hence, Alice's strategy is as follows: if the set $\mathcal{S}'$ is empty, Alice occupies any empty cell of some $s \in \mathcal{S}$. Otherwise, $\mathcal{S}'$ is a nonempty set of animals where any two intersect. Hence, by 2-Hellyness, there exists a cell c that intersects all the animals of $\mathcal{S}'$. Alice will occupy c, preventing Bob from occupying any cell of $\mathcal{S}'$.

With this strategy, Alice makes sure that the set $\mathcal{S}$ never becomes empty (since Bob can never occupy $s' \in \mathcal{S}'$) and the number of Bob's possible moves only decreases after each of Alice's moves. Hence after a nite number of turns, Bob will be unable to play inside the square and Alice will be able to complete a copy of the animal.

The result of Lemma 7 can be extended to an in - nite board, whereby Alice's strategy is to construct a bounded region big enough so that the set $\mathcal{S}$ is nonempty. If she can construct such a region, she can apply the strategy of Lemma 7 by playing only inside this bounded region: From this idea we have the proof of Theorem 6 as follows:

Proof of Theorem 6 (Bounding strategy). Given P, let n and m be the smallest integers such that P is included in $R(n, m)$ (that is $R(n, m)$ is the smallest rectangle that can enclose P). We will construct an $N \quad N$ square region on the board large enough that at least one copy of $R(n, m)$ can be constructed inside (hence so will P). Alice can surround the boundary of the $(N + 2) \quad (N + 2)$ square with at most $4N$ moves (note that the four corners don't need to be occupied). Let I be the interior of the square. Notice that at least $2(N \quad (n \quad 1))(N \quad (m \quad 1))$ copies of $R(n, m)$. can t inside I. Each of Bob's animals stabs at most $(2n \quad 1)(2m \quad 1)+(n+m \quad 1)^2 \leq n^2+m^2+6nm$ copies of $R(n, m)$.

During the (at most) $4N$ rounds during which Alice surrounds the boundary of the square, Bob can stab at most $4N(n^2 + m^2 + 6nm)$ animals of $\mathcal{S}$. Thus, if $2(N \quad n+1)(N \quad m+1) > 4N(n^2+m^2+6nm)$, the set $\mathcal{S}$ will be non-empty even after Alice has completed surrounding the boundary of the square. Because the rst term is quadratic in N and the second is linear, for a su ciently large N the inequality holds.

Corollary 8 *$R(n, m)$ is a non-cannibal (for any $n, m \in \mathbb{N}$).*

For some simple animals, we can construct concrete winning strategies for Alice as follows (For space limitation, their proofs are omitted.):

Lemma 9 *For $S(n)$ and $X(3)$, Alice can win by at most $n^2 + 3$ and 8 moves, respectively.*

4 Concluding remarks

In Harary's generalized tic-tac-toe, some monotone properties hold; these properties include "increasing the size of the board helps Alice" and "increasing the animal helps Bob." However, such properties do not hold for the cannibal animal game, making it deeper and more interesting. We also note that the cannibal property of many other animals is still left unsolved. Among them is the $U(2, 4, 1)$ animal, which we conjecture to be a cannibal, and the squares $S(n)$ in which a cell less than $\lfloor n/4 \rfloor$ units away from the boundary has been removed. On the other hand, it is easy to see that any animal consisting of at most 3 cells is a non-cannibal. We conjecture that all 4-cell-animals are also non-cannibals, and consequently, the 5-cell-animal $U(2, 3, 1)$ would be the smallest cannibal.

Acknowledgments

We are deeply grateful to Professor Ferran Hurtado of Universitat Politècnica de Catalunya for his valuable comments during discussions.

References

[1] E. R. Berlekamp, J. H. Conway, and R. K. Guy, Winning Ways for Your Mathematical Plays, Vol. 3, Second Edition, A K Peters, Massachusetts, 2003.

[2] E. Fisher and N. Seiben, Rectangular polyomino set weak (1, 2)-achievement games, Theoretical Computer Science, **409**, 2008, pp. 333–340.

[3] M. Gardner, Generalized Ticktacktoe, in: Fractal Music, Hypercards and More..., W. H. Freeman and Company, NY, 1992, pp. 202–213.

[4] J.E. Goodman and J. O'Rourke, Handbook of Discrete and Computational Geometry, CRC Press, 1997.

[5] F. Harary, Achieving the skinny animal, Eureka, **42**, 1982, pp. 8–14. Errata in Edition Number 42, **43**, 1983, pp. 5–6.

[6] F. Harary and H. Harborth, Achievemt and avoidance games with triangular animals, J. Recreational Mathematics, **18(2)**, 1985–86, pp. 110–115.

[7] H. Harborth and M. Seemann, Handicap achievement for squares, J. Combin. Math. Combin. Comput., **46**, 2003, pp. 47–52.

[8] H. Ito and H. Miyagawa, Snaky is a winner with one handicap, Abstracts of HERCMA 2007, Sept. 20–22, Athens Univ. of Economics and Business, Athens, Greece, 2007, pp. 25–26.

[9] J. Beck, Combinatorial Games — Tic-Tac-Toe Theory, Cambridge Univ. Press, 2008.

Data Imprecision under λ -Geometry: Finding the Largest Axis-Aligned Bounding Box

Mansoor Davoodi* Payam Khanteimouri* Farnaz Sheikhi* Ali Mohades*

Abstract

In this paper we introduce a new model for handling imprecision in the input data of a geometric problem. The proposed model, which is called λ-*geometry*, is a generalization of region based models to handle dynamic imprecision. Further, we study the problem of finding the largest area axis-aligned bounding box of a set of n imprecise points under the model. We propose an $O\big(n(\log n + \log k + m)\big)$ time algorithm to solve this problem, where k is the maximum complexity of the regions representing imprecise points, and m is the maximum number of corner defining functions when the largest axis-aligned bounding box is defined by points on its corners.

1 Introduction

Classical computational geometry focuses mainly on problems whose input data has precise location. Although these problems have their own interest, they are not very close to what we face in the real world. In the real world, input data is collected by using devices with limited precision. Therefore, imprecision in the location of input data is inevitable. In order to take imprecision into account, several models have been proposed. Region based models are the most common types [5]. Once a model can represent imprecise points, it can be generalized to represent imprecise lines and polygons as well. So, the initial matter is how to model an imprecise point. In a region based model, location of each imprecise point is assumed to be anywhere in a predefined region. The model of ε-geometry [3] is of the earliest models of this type. In this model each point can have an imprecision up to ε in any direction, and as a result regions are assumed to be disks. Other regions such as segments, rectangles, and convex polygons have also been proposed to represent imprecision. Further, several computational geometry problems have been studied under region based models, such as finding the largest/smallest area axis-aligned bounding box [6], and convex hull [5]. A thorough study is presented in [5].

*Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology, {mdmonfared,p.khanteimouri,f.sheikhi,mohades}@aut.ac.ir

Besides solving computational geometry problems under a specific model of imprecision, developing more general and practical models for imprecision is of great importance. Proposing a practical model that deals with dependencies among imprecise input data is of the recent work done in this field [4].

As reviewed above, region based models have a static view on the level of imprecision. That is once the regions that represent imprecision are known; they stay unchanged throughout the algorithm. However, level of imprecision can vary continuously according to circumstances such as changes in the precision of measurement. So, in this paper we introduce a new model that can handle dynamic level of imprecision. Let λ represent the level of imprecision, where $\lambda \in [0, 1]$. Our proposed model which is called λ-*geometry* is a generalization of region based models that allows regions to be growing or shrinking according to increase or decrease in λ in a monotone continuous manner. This model also provides an inherent dependency in the level of imprecision of the input data.

The paper is organized as follows. In Section 2, we define λ-geometry more precisely. Later, in section 3 we show how the problem of finding the maximum area axis-aligned bounding box of a set of imprecise points can be solved under the model of λ-geometry.

2 The model of λ-Geometry

We begin introducing the model of λ-geometry by defining an imprecise point in it. An imprecise point p in this model is defined as $p = \big(\overline{p}, M, \lambda\big)$, where $\overline{p}$ is the *exact value* of the imprecise point p, $M = [v_1, v_2, ..., v_k]_{2\times k}$ is the *imprecision matrix* in which each vector v_i defines the maximum imprecision in its direction, and the parameter λ shows the *imprecision level* for each v_i. So, for any $\lambda \in [0, 1]$, a region is considered for modeling an imprecise point. This region, which includes all possible instances of a point p, is the convex hull of points defined by the sum of the vectors λv_i and $\overline{p}$. Fig.1(a) illustrates a sample imprecise point p for two different λs, $\lambda_1 = 2/3$ and $\lambda_2 = 1$, where $\overline{p} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$ and $M = \begin{bmatrix} 2 & -2 & -3 \\ 0 & 3 & -2 \end{bmatrix}$.

It is obvious that for $\lambda = 1$ an imprecise point p is the convex hull of the points inducing by the vectors of imprecision matrix M, while for $\lambda = 0$ this imprecise

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

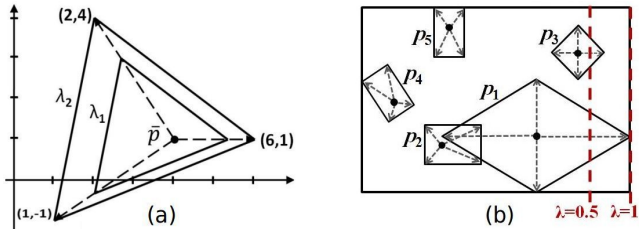


Figure 1: (a) An imprecise point p for $\lambda_1 = 2/3$ and $\lambda_2 = 1$. (b) The AABB constructed with extreme imprecise points in each of the four axis-aligned directions. For $\lambda = 1$, p_1 is the rightmost, while for $\lambda = 0.5$, p_3 is the rightmost imprecise point.

point is just the exact point $\overline{p}$ [1].

Now, we focus on algebraic definition of an instance of an imprecise point, $p(\lambda, \gamma)$:

$$p(\lambda,\gamma) \stackrel{\text{def}}{=} \overline{p} + \lambda.M\gamma \ ; \quad \gamma = [\gamma_1, \gamma_2, \cdots, \gamma_k]^T \ ,$$
$$, 0 \le \gamma_i \le 1, \quad \sum_{i=1}^{k} \gamma_i \le 1.$$

In this definition γ is an arbitrary vector that defines an instance $p(\lambda, \gamma)$ to be anywhere in the convex hull mentioned earlier. So, an imprecise point, $p(\lambda)$, is a region defined by

$$p(\lambda) \stackrel{\text{def}}{=} \big\{p(\lambda,\gamma) \mid \forall\gamma \ ; \ 0 \le \gamma_i \le 1, \sum_{i=1}^{k} \gamma_i \le 1\big\}.$$

In many models of imprecise data, a line defined by two imprecise points is the union of lines that passes through the both corresponding regions [5]. Similarly, a line in the model of λ-geometry is defined as follows:

$$L_{p,q}(\lambda) \stackrel{\text{def}}{=} \big\{L(p', q') \mid p', q' \text{ are instances of } p(\lambda), q(\lambda)\big\}.$$

This definition can be generalized for defining imprecise polygons.

3 Largest Axis-Aligned Bounding Box

Given a set P of n points in the plane, the axis-aligned bounding box (AABB) of P is the smallest axis-aligned rectangle that contains P. The key step to find the AABB of P is to find the extreme point of P in each of the four axis-aligned directions. In an imprecise context when imprecise points are modeled by regions, choosing an instance from each imprecise point results in an AABB. So in this case, the problem is where to choose an instance from each imprecise point to make the area of AABB of these instances maximized for example, (*the largest AABB problem*). From now on we use AABB to refer to the largest AABB.

In the model of λ-geometry, since regions are growing (or shrinking) with the rate of λ, it is possible for each region to be extreme in an axis-aligned direction for a specific $\lambda \in [0, 1]$, see Fig.1(b). So, to find the AABB in the model of λ-geometry, the first step is to break the interval of $[0, 1]$ in some subintervals for which the four extreme regions stay unchanged. Thus, the output will be as follows: $(\lambda_{i-1}, \lambda_i, f_{Ri}, f_{Li}, f_{Ti}, f_{Bi})$, such that $1 \le i \le n'$, $\lambda_i \in [0, 1]$, $\lambda_i < \lambda_{i+1}$, and f_{Ri}, f_{Li}, f_{Ti} and f_{Bi} are functions defining respectively the right, left, top and bottom side of the AABB in the interval of $[\lambda_{i-1}, \lambda_i]$. Next we explain how to find these values.

Consider the imprecise point $p_i(\lambda) = (\overline{p_i}, M_i, \lambda)$, where $\overline{p_i} = (\overline{x_i}, \overline{y_i})$. The rightmost point of $p_i(\lambda)$ is defined by the vector of $v_{iR} = (a_{iR}, b_{iR})$ which is the rightmost vector of M_i. Thus, the rightmost point of $p_i(\lambda)$ can be defined by a linear function $R_i(\lambda) = \overline{x_i} + \lambda a_{iR}$. We call $R_i(\lambda)$ the *rightmost defining function* of $p_i(\lambda)$. Since $p_i(\lambda)$ is a convex polygon, a_{iR} can be found in $O(\log k_i)$ time, where $k_i = |M_i|$ and $|M_i|$ denotes the number of columns in the matrix M_i.

Given a set of imprecise points, let $R = \{R_1, R_2, \cdots, R_n\}$ be the set of all rightmost defining functions. So, the right side of the AABB can be determined by the upper envelope of R. Functions of R are totally defined in $[0, 1]$, and each pair of these functions intersects in at most one point. Thus, the upper envelope of R has at most n breakpoints, and it can be computed in $O(n \log n)$ time [7]. The upper envelopes of the left, top, and bottommost defining functions can be computed similarly. So after computing these four upper envelopes, the union of their breakpoints constructs the critical λs of this step. Let $k = \max_{1 \le i \le n} k_i$. Then the algorithm proposed to find these critical λs runs in $O(n \log k + n \log n)$ time. When k is polynomial in n, we prove this algorithm is optimal.

Lemma 1 *Given a set of n imprecise points in the model of λ-geometry, finding all the rightmost defining functions for the largest AABB problem requires $\Omega(n \log n)$ computations in the worst case.*

Proof. The proof is by a reduction from the sorting problem. Let $S = \{s_1, s_2, \cdots, s_n\}$ be a set of n positive real numbers, and let d be a number greater than the biggest difference of numbers in S. We convert each number s_i to an imprecise point $p_i = (\overline{p_i}, M_i, \lambda)$ with a symmetric rhomboid as its region. We define the rightmost vector of p_i to be $R_i = m_i\lambda - m_i s_i$, where $m_i = s_i/(d - s_i)$. Based on the formula of the rightmost defining function, R_i can be seen as the vector of $V_{iR} = \begin{bmatrix} m_i \\ 0 \end{bmatrix}$ which starts at the point $\overline{p_i} = (-m_i s_i, 0)$. Similarly, we define the left, top and bottom vectors and finally we obtain the imprecision matrix $M = \begin{bmatrix} m_i & 0 & -m_i & 0 \\ 0 & m_i & 0 & -m_i \end{bmatrix}$. We claim that the order in which the rightmost defining functions of the constructed imprecise points appear on the upper envelope is the same as the order of sorted numbers.

[1] Note that since we define p by the concept of convex hull, it is possible for a vector to be ineffective. But without loss of generality, we assume that all vectors are effective.

To prove this it is sufficient to show that for any three positive numbers s_i, s_j, s_k, where $s_i < s_j < s_k$, $R_i(\lambda)$ appears before $R_j(\lambda)$ on the upper envelope, and if x_{ij} (resp. x_{ik}) is the intersection point of $R_i(\lambda)$ and $R_j(\lambda)$ $\big($resp. $R_i(\lambda)$ and $R_k(\lambda)\big)$, then $x_{ij} < x_{ik}$. This is a straightforward result. □

In finding critical λs and the corresponding right, left, top and bottommost defining functions there is still one issue left to handle, and that is we can not choose more than one instance from each imprecise point. For a fixed value of λ, Fig.1(b) illustrates a situation where the right and the bottom side of the AABB is defined by two different instances of an imprecise point $p_1(\lambda)$. To handle this situation either we should take just the rightmost or just the bottommost instance of $p_1(\lambda)$, or we should take an instance of $p_1(\lambda)$ that plays these roles together that is the instance taken should be the lower-right corner of the AABB. The final choice is the one that leads to a larger area of AABB. Next we explain how to handle this issue in the model of λ-geometry.

For a fixed value of λ, the AABB of a set of n imprecise points modeled by regions can be determined by testing only the four farthest imprecise points in each of the four axis-aligned directions [6]. Further, let $F = \{f_1, f_2, \cdots, f_n\}$ be an arrangement of n lines in the plane and let c be a constant. It is known that the c^{th}-upper level of F, denoted by E_F^c, has $O(n)$ breakpoints and it can be computed in $O(n \log n)$ time [2]. So, in the model of λ-geometry to find the AABB determined by four instances of different imprecise points we do the following. First we compute the set of rightmost, the set of leftmost, the set of topmost and the set of bottommost defining functions. Let R, L, T, and B denote these sets respectively. Then, for each of these sets we find c^{th}-upper level for $c = 1, 2, 3, 4$. Let E_R^c, E_L^c, E_T^c and E_B^c denote these upper levels for the sets R, L, T, and B respectively. Now, we define the set of critical λs as:

$$\Lambda = \big\{\lambda | \lambda \text{ is a break point for } E_R^c \text{ or } E_L^c \text{ or } E_T^c \text{ or } E_B^c \text{ for } c = 1, 2, 3, 4\big\}$$

Any two consecutive members of the ordered set Λ construct an interval in which none of the four upper levels changes. So, in each interval constructed if the first upper level of each set is different from the first upper levels of the other sets, we report this interval and its corresponding upper levels (in this case we are sure that four different imprecise points have constructed the AABB). Otherwise, if the first upper levels of some sets are equal, we take the first upper level of a set and test the second, the third, and finally in the worst case the forth upper levels of the remaining sets, and we select the one that leads to a larger area of AABB. So, by the approach explained we can find the AABB that is determined by four instances of different imprecise points. Further, when one or two instances play the role of corners of the AABB, three or two instances of different imprecise points can also determine the AABB. So, we should compute the AABB of these situations as well, and at the end report the largest AABB we have found over all situations. The case of three instances occurs when one corner of the AABB is constructed by two equal upper levels (for example when $E_L^1 \neq E_T^1$ and $E_R^1 = E_B^1$). Further, the case of two instances occurs when two opposite corners of the AABB are constructed by equal upper levels (for example when $E_L^1 = E_T^1$ and $E_R^1 = E_B^1$). So, the final step of the algorithm is how to find the AABB of these two cases. Next we show how to handle this.

Let $p(\lambda)$ be an imprecise point whose instance is a corner of the AABB. Obviously this instance should be taken from the boundary of $p(\lambda)$. Thus, the boundary of $p(\lambda)$, and more precisely, a specific part of this boundary called a chain is important to us. Assume that vertices of the boundaries of the imprecise points are in a clockwise order. We call a chain *convex* (resp. *concave*) if it is xy-monotone and it lies above (resp. below) any line that passes through its segments.

Lemma 2 *Let $C_p = \{p_1, p_2, \cdots, p_k\}$ be a concave chain in the second quadrant of the coordinate system, while $C_q = \{q_1, q_2, \cdots, q_k\}$ is a convex chain in the forth quadrant. Then the AABB whose corners lie on C_p and C_q can be computed in $O(\log k)$ time.*

Proof. It is known that the largest inscribed isothetic rectangle inside a k-sided polygon can be computed by an $O(\log k)$ time algorithm [1]. We can treat the two chains' problem as a special case of this algorithm. Thus, the AABB whose corners lie on C_p and C_q can be computed in $O(\log k)$ time as well. □

Lemma 3 *Let $Cor(\lambda) = (\lambda a + a', \lambda b + b')$ represent the upper-left corner of the AABB and let $q = (\overline{q}, M_q, \lambda)$ be the rightmost and bottommost imprecise point. Then, the AABB can be computed in $O(\log k + m)$ time, where $k = |M_q|$ and m is the number of functions defining the lower-right corner of the AABB, while λ decreases from 1 to 0.*

The proof is omitted due to space limitations. Since lemma 3 is a special case of lemma 4, we refer readers to see the proof of lemma 4.

So far we have shown how to find the AABB constructed by four instances, and the AABB constructed by three instances of imprecise points. The following lemma shows how to find the AABB constructed by two instances of imprecise points.

Lemma 4 *Let $p = (\overline{p}, M_p, \lambda)$ be the leftmost and topmost, and let $q = (\overline{q}, M_q, \lambda)$ be the rightmost and bottommost imprecise points. Then the AABB can be computed in $O(\log k + m)$ time, where $k =$*

$\max\big(|M_p|, |M_q|\big)$ *and* m *is the number of functions defining corners, while* λ *decreases from* 1 *to* 0.

Proof. It is known that in the case of two segments, at least one corner of the AABB lies on endpoints of these segments [1]. So, for any fixed value of λ, at least one vector of M_p and one edge of M_q (or vice versa) define the AABB. This is still true while points are growing or shrinking in the model of λ-geometry. This fact helps us to find the optimal path for each of two corners of the AABB. To do so, it is sufficient to find critical λs for which the vector or the segment that defines the AABB changes. Let $l'\big(a(\lambda), s(\lambda)\big)$ denote the optimal path for the point $a(\lambda)$ and the segment $s(\lambda)$. The parametric equation of $l'\big(a(\lambda), s(\lambda)\big)$ can be obtained by computing the area function and its first derivative [2]. First for $\lambda = 1$ we compute two corners of the AABB by using lemma 2. These corners are taken from p and q, and are denoted by c_p and c_q respectively. Set $\lambda_c = 1$. According to the position of these corners two cases arise:

Case1: The corner c_p lies on a vertex of $p(\lambda)$, while the corner c_q lies on a segment of $q(\lambda)$(or vice versa). We denote this vertex of $p(\lambda)$ by p_i, and this segment of $q(\lambda)$ by $s_j^q = \overline{q_j q_{j+1}}$ (see Fig.2 for $0.7 \le \lambda \le 1$ and $0.2 \le \lambda \le 0.3$). In this case, we take vectors v_j and v_{j+1} of M_q, and find their intersection with line $l'(c_p, s_j^q)$. Note that only one of these vectors intersects $l'(c_p, s_j^q)$. Then we compute the value of λ corresponding to this intersection, and denote it by λ_q. Further, we take the vector v_i of M_p and find its intersection with $l'(v_j, s_i^p)$ and $l'(v_j, s_{i+1}^p)$ (just one of these lines intersects v_i). Let λ_p be the value of λ corresponding to this intersection. Set $\lambda_n = \max(\lambda_p, \lambda_q)$ and report two functions v_i and $l'(c_p, s_j^q)$ as two corners' paths in $[\lambda_n, \lambda_c]$. If $\lambda_n \le 0$ the algorithm is finished. Otherwise, set $\lambda_c = \lambda_n$ and go to case 2.

Case2: Both corners c_p and c_q lie on vertices of $p(\lambda)$ and $q(\lambda)$. Let p_i denote the corresponding vertex of $p(\lambda)$, and let q_j denote the corresponding vertex of $q(\lambda)$ (see Fig. 2 for $0.3 \le \lambda \le 0.7$ or $0 \le \lambda \le 0.2$). In this case, we take vector v_j of M_q, and find its intersection with $l'(v_i, s_j^q)$ and $l'(v_i, s_{j+1}^q)$. Let λ_q denote the value of λ corresponding to this intersection. Further, we take vector v_i of M_p, and compute its intersection with $l'(v_j, s_i^p)$ and $l'(v_j, s_{i+1}^p)$. Let λ_p denote the value of corresponding λ. Set $\lambda_n = \max(\lambda_p, \lambda_q)$ and report two functions v_i and v_i as two corners' paths in $[\lambda_n, \lambda_c]$. If $\lambda_n \le 0$ the algorithm is finished. Otherwise, set $\lambda_c = \lambda_n$ and go to case 1.

In each iteration, the algorithm reports an interval and two functions defining the optimal paths of the upper left and lower right corners within this interval. Let m denote the total number of corner defining functions in $[0, 1]$. Then the algorithm iterates m times.

[2]Note that while λ decreses from 1 to 0, $a(\lambda)$ is a segment while $s(\lambda)$ is a triangle.

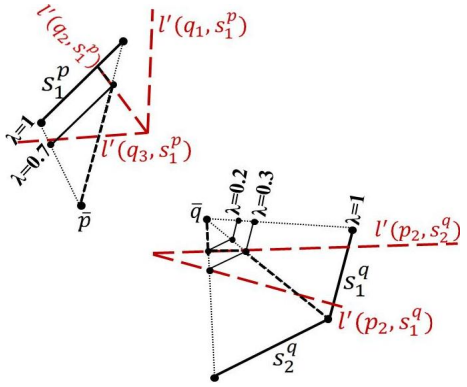


Figure 2: Concepts used in finding corner defining functions. Short dashes show the optimal paths for the corners.

Finding the first optimal corners (for $\lambda = 1$) takes $O(\log k)$ time by using lemma 2, but since we have these corners in the following iterations, each of the following iterations takes constant time. Therefore, the total time complexity is $O(\log k + m)$. Note that $m = O(k)$. □

Thus, in the model of λ-geometry we have obtained the following results:

Theorem 5 *Given a set of* n *imprecise points under the model of* λ*-geometry, the largest AABB problem can be solved in* $O\big(n(\log n + \log k + m)\big)$ *time, where* k *is the maximum complexity of regions representing imprecision, and* m *is the maximum number of corner defining functions.*

Note that when $k = O(\log n)$, the largest AABB problem under the model of λ-geometry can be solved in the optimal time $\Theta(n \log n)$.

References

[1] H. Alt, D. Hsu, J. Snoeyink. Computing the largest inscribed isothetic rectangle. In *Proc. 7th Canadian Conf. Comput. Geom.,* pp. 67-72, 1995.

[2] H. Everett, J. M. Robert, M. van Kreveld. An Optimal Algorithm for Computing ($\le k$)-Levels, with Application to Separation and Transversal Problems. *Internat. J. Comput. Geom. Appl.*, 6, pp. 247-261, 1996.

[3] L. Guibas, D. Salesin, J. Stolfi. Epsilon geometry: Building robust algorithms for imprecise computations. In *Proc. 5th ACM Annual Symposium on Comput. Geom.,* pp. 208-217, 1989.

[4] Y. Myers, L. Joskowicz. Uncertain geometry with dependencies. In *Proc. 14th ACM Symposium on Solid and Physical Modeling,* pp. 159-164, 2010.

[5] M. Löffler. *Data Imprecision in Computational Geometry.* PhD Thesis, Utrecht University, 2009.

[6] M. Löffler, M. van Kreveld. Largest Bounding Box, Smallest Diameter, and Related Problems on Imprecise Points. In *Proc. WADS,* pp. 446-457, 2007.

[7] M. Sharir, P.K. Agarwal. *Davenport-Schinzel Sequences and their Geometric Applications.* Cambridge University Press, 1995.

A Proof of the Oja-depth Conjecture in the Plane

Nabil H. Mustafa* Hans Raj Tiwary† Daniel Werner‡

Abstract

Given a set P of n points in the plane, the *Oja-depth* of a point $x \in \mathbb{R}^2$ is defined to be the sum of the areas of all triangles defined by x and two points from P, normalized by the area of convex-hull of P. The Oja-depth of P is the minimum Oja-depth of any point in $\mathbb{R}^2$. The Oja-depth conjecture states that any set P of n points in the plane has Oja-depth at most $n^2/9$ (this would be optimal as there are examples where it is not possible to do better). We present a proof of this conjecture.

We also improve the previously best bounds for all $\mathbb{R}^d$, $d \geq 3$, via a different, more combinatorial technique.

1 Introduction

We first present some examples of the several different versions of data-depth that have been studied.

The *location-depth* of a point x is the minimum number of points of P lying in any halfspace containing x [11, 20, 19]. The Center-point Theorem [9] asserts that there is always a point of location-depth at least $n/(d+1)$, and that this is the best possible. The point with the highest location-depth w.r.t. to a point-set P is called the *Tukey-median* of P. The corresponding computational question of finding the Tukey-median of a point-set has been studied extensively, and an optimal algorithm with running time $O(n \log n)$ is known in $\mathbb{R}^2$ [7].

The *simplicial-depth* [13] of a point x and a set P is the number of simplices spanned by P that contain x. The First Selection Lemma [14] asserts that there always exists a point with simplicial-depth at least $c_d \cdot n^{d+1}$, where $c > 0$ is a constant depending only d. The optimal value of c_d is known only for $d = 2$, where $c_2 = 1/27$ [5]. For c_3 is still open, though it has been the subject of a flurry of work recently [3, 6, 10]. The current-best algorithm computes the point with maximum simplicial-depth in time $O(n^4 \log n)$ [1].

The L_1 depth, proposed by Weber in 1909, is defined to be the sum of the distances of x to the n input points. It is known that the point with the lowest such depth is unique in $\mathbb{R}^2$.

Oja-depth. In this paper, we study another well-known measure called the *Oja depth* of a point-set. Given a set P of n points in $\mathbb{R}^d$, the *Oja-depth* (first proposed by Oja [16] in 1983) of a point $x \in \mathbb{R}^d$ w.r.t. P is defined to be the sum of the volumes of all d-simplices spanned by x and d other points of P. Formally, given a set $Q \subset \mathbb{R}^d$, let $\text{conv}(Q)$ denote the convex-hull of Q, and let $\text{vol}(Q)$ denote its d-dimensional volume. Then,

$$\text{Oja-depth}(x) = \sum_{y_1,\ldots,y_d \in \binom{P}{d}} \frac{\text{vol}(\text{conv}(x, y_1, \ldots, y_d))}{\text{vol}(\text{conv}(P))}$$

The Oja-depth of P is the minimum Oja-depth over all $x \in \mathbb{R}^d$. From now onwards, w.l.o.g., assume that $\text{vol}(\text{conv}(P)) = 1$.

Known bounds. First we note that

$$\left(\frac{n}{d+1}\right)^d \leq \text{Oja-depth}(P) \leq \binom{n}{d}.$$

For the upper-bound, observe that any d-simplex spanned by points inside the convex-hull of P can have volume at most 1, and so a trivial upper-bound for Oja-depth of any $P \subset \mathbb{R}^d$ is $\binom{n}{d}$, achieved by picking any $x \in \text{conv}(P)$. For the lower-bound, construct P by placing $n/(d+1)$ points at each of the $d+1$ vertices of a unit-volume simplex in $\mathbb{R}^d$.

The conjecture [8] states that this lower bound is tight:

Conjecture 1 *Oja-depth*$(P) \leq (\frac{n}{d+1})^d$ *for any* $P \subset \mathbb{R}^d$ *of* n *points.*

The current-best upper-bound [8] is that the Oja-depth of any set of n points is at most $\binom{n}{d}/(d+1)$. In particular, for $d = 2$, this gives $n^2/6$.

The Oja-depth conjecture states the existence of a low-depth point, but given P, computing the *lowest*-depth point is also an interesting problem. In $\mathbb{R}^2$, Rousseeuw and Ruts [18] presented a straightforward $O(n^5 \log n)$ time algorithm for computing the lowest-depth point, which was improved to the current-best

*Dept. of Computer Science, LUMS, Pakistan. nabil@lums.edu.pk

†Département de Mathématique, Université Libre de Bruxelles, Belgium, hans.raj.tiwary@ulb.ac.be

‡Institut für Informatik, Freie Univ. Berlin, Germany. daniel.werner@fu-berlin.de - This research was funded by Deutsche Forschungsgemeinschaft within the Research Training Group (Graduiertenkolleg) "Methods for Discrete Structures"

algorithm with running time $O(n \log^3 n)$ [1]. An approximate algorithm utilizing fast rendering systems on current graphics hardware was presented in [12, 15]. For general d, various heuristics for computing points with low Oja-depth were given by Ronkainen, Oja and Orponen [17].

Our results. In Section 2, we present our main theorem, which completely resolves the conjecture for the planar case.

Theorem 1 *Every set P of n points in $\mathbb{R}^2$ has Oja-depth at most $\frac{n^2}{9}$. Furthermore, such a point can be computed in $O(n \log n)$ time.*

In Section 3, using completely different (and more combinatorial) techniques for higher dimensions, we also prove the following:

Theorem 2 *Every set P of n points in $\mathbb{R}^d$, $d \geq 3$, has Oja-depth at most $\frac{2n^d}{2^d d!} - \frac{2d}{(d+1)^2(d+1)!}\binom{n}{d} + O(n^{d-1})$.*

This improves the previously best bounds by an order of magnitude.

2 The optimal bound for the plane

We now come to prove the optimal bound for $\mathbb{R}^2$. First, let us give some basic definitions. The *center of mass* or *centroid* of a convex set X is defined as

$$c(X) = \frac{\int_{x \in X} x \, dx}{\text{area}\,(X)}.$$

For a discrete point set P, the center of mass is simply defined as the center of mass of the convex hull of P. When we talk about the *centroid of P*, we refer to the center of mass of the convex hull and hope the reader does not confuse this with the discrete centroid $\sum p/|P|$. In what follows, we will bound the Oja-depth of the centroid of a set, and show that it is worst-case optimal. Our proof will rely on the following two Lemmas.

Lemma 3 *[Winternitz [4]] Every line through the centroid of a convex object has at most $\frac{5}{9}$ of the total area on either side.*

Lemma 4 *[8] Let P be a convex object with unit area and let c be its center of mass. Then every simplex inside P which has c as a vertex has area at most $\frac{1}{3}$.*

To simplify matters, we will use the following proposition.

Proposition 5 *If we project an interior point $p \in P$ radially outwards from the centroid c to the boundary of the convex hull, the Oja-depth of the point c does not decrease.*

Proof. First, observe that the center of mass does not change. It suffices to show that every triangle that has p as one of its vertices increases its area. Let $T := \Delta(c, p, q)$ be any triangle. The area of T is $\frac{1}{2}\|c - p\| \cdot h$, where h is the height of T with respect to $p - c$. If we move p radially outwards to a point p', h does not change, but $\|c - p'\| > \|c - p\|$. □

This implies that in order to prove an upper bound, we can assume that all points lie on the convex hull.

From now on, let P be a set of points, and let $c := c(\text{conv}(P))$ denote its center of mass as defined above. Further, let $p_1, \ldots, p_n$ denote the points sorted clockwise by angle from c. We define the *distance* of two points as the difference of their position in this order (modulo n). A triangle that is formed by c and two points at distance i is called an *i-triangle*, or *triangle of type i*. Observe that for each i, $1 \leq i < \lfloor n/2 \rfloor$, there are exactly n triangles of type i. Further, if n is even, then there are $n/2$ triangles of type $\lfloor n/2 \rfloor$, otherwise there are n. These constitute all possible triangles.

Let $C \subseteq P$, and let $\mathcal{C}$ be they boundary of the convex hull of C. This will be called a *cycle*. The length of a cycle is simply the number of elements in C. A cycle $\mathcal{C}$ of length i *induces i triangles* that arise by taking all the triangle formed by an edge in $\mathcal{C}$ and the center of mass c (of $\text{conv}(P)$). The area induced by $\mathcal{C}$ is the sum of areas of these i triangles.

The triangles induced by the entire set P form a partition of $\text{conv}(P)$. Thus, Lemma 5 implies the following:

Corollary 6 *The total area of all triangles of type 1 is exactly 1.*

The following shows that we can generalize this Lemma, i.e., that we can bound the total area induced by *any* cycle.

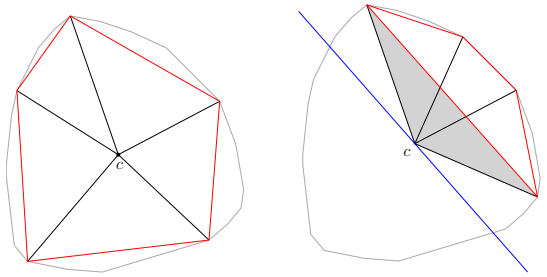
Lemma 7 *Let $\mathcal{C}$ be a cycle. Then $\mathcal{C}$ induces a total area of at most 1.*

Proof. We distinguish two cases.

Case 1: The centroid lies in the convex hull of $\mathcal{C}$. In this case, all triangles are disjoint, so the area is at most 1. See Fig. 1(a).

Case 2: The centroid does not lie in the convex hull of $\mathcal{C}$. By the Separation Theorem [14], there is a line through c that contains all the triangles. Then we can remove one triangle to get a set of disjoint triangles, namely the one induced by the pair $\{p_{i_j}, p_{i_{j+1}}\}$ that has c on the left side. By Lemma 3, the area of the remaining triangles can thus be at most 5/9. By Lemma 4, the removed triangle has an area of at most 1/3. Thus, the total area is at most 8/9. See Fig. 1(b). Here, the gray triangle can be removed to get a set of disjoint triangles.

□



(a) Case 1 (b) Case 2

Figure 1: The two cases

We now prove the crucial lemma, which is a general version of Corollary 6.

Lemma 8 *The total area of all triangles of type i is at most i.*

Proof. We will proceed as follows: For fixed i, we will create n cycles. Each cycle will consist of one triangle of type i, and $n-i$ triangles of type 1, multiplicities counted. We then determine the total area of these cycles and subtract the area of all 1-triangles. This will give the desired result.

Let $p_1, \ldots, p_n$ be the points ordered by angles from the centroid c. Let $\mathcal{C}_j$ be the cycle consisting of the $n-i+1$ points $P - \{p_{i+1 \mod n}, \ldots, p_{i+j-1 \mod n}\}$. This is a cycle that consists of one triangle of type i, namely the one starting a p_j, and $n-i$ triangles of type 1.

By Lemma 7, every cycle $\mathcal{C}_j$ induces an area of at most 1. If we sum up the areas of all n cycles $\mathcal{C}_j$, $1 \le j \le n$, we thus get an area of at most n.

We now determine how often we have counted each triangle. Each i-triangle is counted exactly once. Further, for every cycle we count $n-i$ triangles of type 1. For reasons of symmetry, each 1-triangle is counted equally often. Thus, each is counted *exactly* $n-i$ times over all the cycles. By Corollary 6, their area is *exactly* $n-i$, which we can subtract from n to get the total area of the i-triangles:

$$\begin{aligned}\sum_{i-\Delta\ T} \text{area}(T) &\le n - \left(\sum_{1-\Delta\ T} (n-i)\,\text{area}(T)\right) \\ &= n - (n-i) = i.\end{aligned}$$

This completes the proof. □

Theorem 9 *Let P be any set of points in the plane and c be the centroid of its convex hull. Then the Oja-depth of c is at most $\frac{n^2}{9}$.*

Proof. We will bound the area of the triangles depending on their type. For i-triangles with $1 \le i \le \lfloor n/3 \rfloor$, we will use Lemma 8. For i-triangles with $\lfloor n/3 \rfloor < i \le \lfloor n/2 \rfloor$, this would give us a bound worse than $n/3$, so we will use Lemma 4 for each of these.

By Lemma 8, the sum of the areas of all triangles of type at most $\lfloor n/3 \rfloor$ is at most

$$\sum_{i=1}^{\lfloor n/3 \rfloor} i = \frac{\lfloor n/3 \rfloor (\lfloor n/3 \rfloor + 1)}{2} \le \frac{n^2}{18} + \frac{1}{2}\lfloor n/3 \rfloor .$$

For the remaining triangles, we use Lemma 4 to bound the size of each by 1/3. Thus, in total we get

$$\text{Oja-depth}(P) \le \frac{n^2}{18} + \frac{n(\lfloor n/2 \rfloor - \lfloor n/3 \rfloor)}{3} + \frac{n}{6}.$$

By a simple case distinction, it is easy to see that the lower order term disappears. This finishes the proof. □

3 Higher Dimensions

We now present improved bounds for the Oja-depth problem in dimensions greater than two. Before the main theorem, we need the following two lemmas.

Lemma 10 *Let P be a set of n points in $\mathbb{R}^d$. Let $q \in \mathbb{R}^d$. Then any line l through q intersects at most $f(n,d)$ $(d-1)$-simplices spanned by P, where $f(n,d) = \frac{2n^d}{2^d d!} + O(n^{d-1})$.*

Proof. Project P onto the hyperplane H orthogonal to l to get the point-set P' in $\mathbb{R}^{d-1}$. The line l becomes a point on H, say point p_l. Then l intersects the $(d-1)$-simplex spanned by $\{p_1, \ldots, p_d\}$ if and only if the convex hull of the corresponding points in P' contain the point p_l.

By a result of Barany [2], any point in $\mathbb{R}^d$ is contained in at most

$$\frac{2(n-d)}{n+d+2}\binom{(n+d+2)/2}{d+1} + O(n^d)$$

simplices induced by a point set.

Applying this lemma to P' in $\mathbb{R}^{d-1}$ and simplifying the expression, we get the desired result. □

Lemma 11 *Given any set P of n points in $\mathbb{R}^d$, there exists a point q such that any half-infinite ray from q intersects at least $\frac{2d}{(d+1)^2(d+1)!}\binom{n}{d}$ $(d-1)$-simplices spanned by P.*

Proof. Gromov [10] showed that, given any set P, there exists a point q contained in at least $\frac{2d}{(d+1)(d+1)!}\binom{n}{d+1}$ simplices spanned by P. Now any half-infinite ray from q must intersect exactly one $(d-1)$-dimensional face (which is a $(d-1)$-simplex) of each d-simplex containing q, and each such $(d-1)$-simplex can be counted at most $n-d$ times. □

Theorem 12 *Given any set P of n points in $\mathbb{R}^d$, there exists a point q with Oja-depth at most*

$$B := \frac{2n^d}{2^d d!} - \frac{2d}{(d+1)^2(d+1)!}\binom{n}{d} + O(n^{d-1}).$$

Proof. Let q be the point from Lemma 11. Let $w(r)$ denote the number of simplices spanned by q and d points from P that contain r. In what follows, we will give a bound on $w(r)$, and thus on the Oja-depth of q.

If r is contained in a simplex, then any half-infinite ray $\overrightarrow{qr}$ intersects a $(d-1)$-facet of that simplex. Therefore, $w(r)$ is upper-bounded by the number of $(d-1)$-simplices spanned by P that are intersected by the ray $\overrightarrow{qr}$.

To upper-bound this, note that the ray starting from q but in the opposite direction to the ray $\overrightarrow{qr}$, intersects at least $\frac{2d}{(d+1)^2(d+1)!}\binom{n}{d}$ $(d-1)$-simplices (by Lemma 11). On the other hand, by Lemma 10, the entire line passing through q and r intersects at most $\frac{2n^d}{2^d d!} + O(n^{d-1})$ $(d-1)$-simplices. These two together imply that the ray $\overrightarrow{qr}$ intersects at most B $(d-1)$-simplices, and this is also an upper-bound on $w(r)$. Finally, we have

$$\begin{aligned} \text{Oja-depth}(q, P) &= \int_{\text{conv}(P)} w(x)\, dx \\ &\leq \int_{\text{conv}(P)} B\, dx = B \end{aligned}$$

finishing the proof. □

Acknowledgments. This research was done during the DCG Special Semester in Lausanne. We thank the EPFL and the organizers János Pach and Emo Welzl.

References

[1] G. Aloupis. On computing geometric estimators of location, 2001. M.Sc. Thesis, McGill University.

[2] I. Barany. A generalization of caratheodory's theorem. *Discrete Mathematics*, 40:141–152, 1982.

[3] A. Basit, N. H. Mustafa, S. Ray, and S. Raza. Hitting simplices with points in 3d. *Discrete & Computational Geometry*, 44(3):637–644, 2010.

[4] W. Blaschke. *Vorlesungen uber Differentialgeometrie. II, Affine Differentialgeometrie.* Springer, 1923.

[5] E. Boros and Z. Furedi. The maximal number of covers by the triangles of a given vertex set on the plane. *Geom. Dedicata*, 17:69–77, 1984.

[6] B. Bukh, J. Matousek, and G. Nivasch. Stabbing simplices by points and flats. *Discrete & Computational Geometry*, 43(2):321–338, 2010.

[7] T. M. Chan. An optimal randomized algorithm for maximum tukey depth. In *SODA*, pages 430–436, 2004.

[8] D. Chen, O. Devillers, J. Iacono, S. Langerman, and P. Morin. Oja Medians and Centers of Mass. In *Proceedings of the 22nd CCCG*, pages 147–150, 2010.

[9] J. Eckhoff. Helly, Radon and Carathéodory type theorems. In *Handbook of Convex Geometry*, pages 389–448. North-Holland, 1993.

[10] M. Gromov. Singularities, expanders and topology of maps. part 2: From combinatorics to topology via algebraic isoperimetry. *Geometric and Functional Analysis*, 20:416–526, 2010.

[11] J. Hodges. A bivariate sign test. *Annals of Mathematical Statistics*, 26:523–527, 1955.

[12] S. Krishnan, N. Mustafa, and S. Venkatasubramanian. Statistical data depth and the graphics hardware. In *Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications*, pages 223–250. DIMACS Series, 2006.

[13] R. Liu. A notion of data depth based upon random simplices. *The Annals of Statistics*, 18:405–414, 1990.

[14] J. Matoušek. *Lectures in Discrete Geometry.* Springer-Verlag, New York, NY, 2002.

[15] N. H. Mustafa. *Simplification, Estimation and Classification of Geometric Objects.* PhD thesis, Duke University, Durham, North Carolina, 2004.

[16] H. Oja. Descriptive statistics for multivariate distributions. *Statistics and Probability Letters*, 1:327–332, 1983.

[17] T. Ronkainen, H. Oja, and P. Orponen. Computation of the multivariate oja median. In *R. Dutter and P. Filzmoser.* International Conference on Robust Statistics, 2003.

[18] P. Rousseeuw and I. Ruts. Bivariate location depth. *Applied Statistics*, 45:516–526, 1996.

[19] P. J. Rousseeuw, I. Ruts, and J. W. Tukey. The bagplot: A bivariate boxplot. *The American Statistician*, 53(4):382–387, 1999.

[20] J. Tukey. Mathematics and the picturing of data. In *Proc. of the international congress of mathematicians*, pages 523–531, 1975.

High Quality Surface Mesh Generation for Swept Volumes

Andreas von Dziegielewski * Michael Hemmer†

Abstract

We present a novel and efficient technique to generate a high quality mesh that approximates the outer boundary of a swept volume (SV). Our approach comes with two guarantees. First, the approximation is conservative, i.e. the swept volume is enclosed by the output mesh. Second, the one-sided Hausdorff distance of the generated mesh to the swept volume is upper bounded by a user defined tolerance. Exploiting this tolerance our method produces an anisotropic mesh which nicely adapts to the local complexity of the approximated swept volume boundary. The algorithm is two phased: a initialization phase that generates a conservative voxelization of the swept volume, and the actual mesh generation which is based on CGAL's Delaunay refinement implementation.

1 Introduction

The swept volume is defined as the entity of all points touched by a solid (the generator) under the transformations of either a continuous or discrete trajectory. In the context of this paper, we assume a discrete trajectory and define $\mathcal{SV}$ as the entity obtained by linear interpolating the generator geometry between consecutive time steps, as proposed in [2].

The swept volume plays an important role in computer aided design (CAD), numerically controlled (NC) machining verification, robot analysis and graphical modeling. For safety reasons most applications demand for a conservative approximation of $\mathcal{SV}$, that is, the result must contain $\mathcal{SV}$. At the same time, the result should approximate $\mathcal{SV}$ as close as possible while keeping the complexity of the output low, the latter becoming more and more relevant due to increasing model complexity and high demands concerning error tolerance. Moreover, a practical algorithm should be tolerant to topologically inconsistent input data and should preferably impose no restrictions on the input models.

1.1 Previous and related work

Mathematical formulations describing the swept volume include Jacobian rank deficiency methods, sweep differential equations and envelope theory, for a survey see [1].

Practical approaches, most relying on volumetric SV representations cannot give geometrical guarantees because sharp features can be missed ([7, 6]). In [11] the authors claim to be able to archieve arbitrarily tight error-bounds (although do not regard conservativeness) but unfortunately no timings for a priori error bounds are given. They all produce a highly overtessallated mesh and do not propose any error bound mesh simplification method. Applying error bound simplification (e.g. [10]) to these meshes, a posteriori, will always be limited by the storage needed for the mass of triangles of the original mesh.

Recent approaches [9, 3] are able to produce a conservative approximation. In [3] local culling criteria are applied to generate a superset of the SV boundary that is then blended using a BSP tree. Their output mesh is non-adaptive and possibly highly overtessellated, and further simplification in convex regions would violate conservativeness. The method proposed in [9] relies on special conservative depth buffer voxelization and an error bound mesh simplification phase. The output mesh is adaptive and almost everywhere manifold. It is conservative, but error bounds can not be given for all concave parts of the SV.

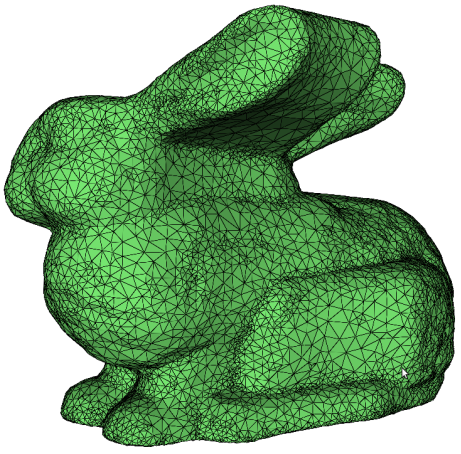


Figure 1: Bunny swept along a trajectory consisting of 12 transformations.

1.2 Our contribution

In contrast to these previous approaches our mesh is build top down, i.e. we start from a coarse mesh and

*Institut für Informatik, Johannes Gutenberg Universität Mainz, dziegiel@uni-mainz.de

†School for Computer Science, Tel Aviv University, mhsaar@google.mail.com

apply local refinement until we are guaranteed a conservative and error bound approximation of the outer $\mathcal{SV}$ boundary. More precisely, we guarantee that the one-sided Hausdorff distance to $\mathcal{SV}$ does not exceed the user defined tolerance δ. In addition the resulting anisotropic mesh is of high quality while keeping the complexity relatively low, that is: produced triangles obey quality constrains such as lower bounds on smallest angles while mesh density adapts to the local complexity of $\mathcal{SV}$ as far as it is required by δ. The algorithm is two phased: a initialization phase that generates a conservative voxelization of the swept volume, and the actual mesh generation which is based on CGAL's Delaunay refinement implementation [8].

The Delaunay refiment and the general scheme of the approach are discussed in Section 2. More details of the initialization phase, are then discussed in Section 3. Implementation details and some preliminary results are presented in Section 4.

2 Overview

In order to generate the output mesh we apply Delaunay refinement which is known to be one of the most powerful techniques in the field of mesh generation and surface approximation. More precisely, our algorithm utilizes CGAL's frame work for 3D mesh generation [8] which we briefly review next.

Starting from an initial points set on the surface of the to be meshed domain $\mathcal{D}$, the process maintains a Delaunay triangulation of this point set. Thereby, it classifies each tetrahedron as interior or exterior according to the position of the center of its circumscribing ball. A triangle is said to belong to the surface if the classification of the two neighboring tetrahedra differ. Such a surface facet f can be refined by inducing the intersection point of its Voronoi edge (the dual of f) with $\partial\mathcal{D}$.[1] The refinement process now successively refines those boundary facets that are classified as bad facets, for instance, triangles that are considered two large or whose minimal angle is too small. The latter criterion ensures well formed triangles in the resulting mesh.

The main idea of our approach is to add another criterion that classifies a boundary facet as bad if: (a) the facet intersects $\mathcal{SV}$ or (b) the one-sided Hausdorff-distance of the facet to $\mathcal{SV}$ is too large. Note that part (a) implies that we can not place intersection points directly on $\partial\mathcal{SV}$ since boundary facets would always intersect $\mathcal{SV}$ in regions where $\mathcal{SV}$ is locally convex. Thus the idea is to place intersections points on some offset of $\mathcal{SV}$ at about half of the user defined tolerance. This way it is guaranteed that boundary facets eventually fulfill both conditions which in turn guarantees termination of the generation process.

[1] On the duality of the Delaunay triangulation and the Voronoi diagram see for instance [4]

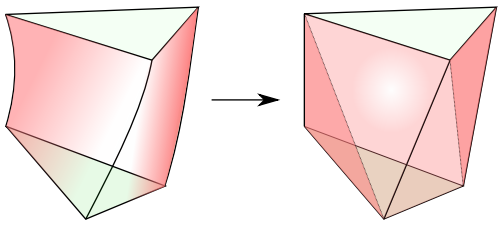


Figure 2: Sweeping a triangle yields a prism with three ruled surfaces, each of which is approximated and tessellated by inserting the diagonal with the lower dihedral angle.

The approach is two phased: In a initialization phase we generate a conservative and sufficiently precise voxelization $\mathcal{V}_0$ of $\mathcal{SV}$. In addition we compute two offsets $\mathcal{V}_1$ and $\mathcal{V}_2$ by successively adding an additional layer of voxels to $\mathcal{V}_0$. In the second phase, the mesh generation phase, we set $\mathcal{D} = \mathcal{V}_1$. Intersection points of Voronoi edges with $\partial\mathcal{V}_1$ are computed using bisection. In order to classify a facet, the facet is voxelized. The facet is considered as bad if one of its voxels is contained in $\mathcal{V}_0$ (ensuring (a)) or outside of $\mathcal{V}_2$ (ensuring (b)).

Thus the core of our approach is an efficient generation of $\mathcal{V}_0$ and its offsets, which is discussed next.

3 Initialization Phase

In order to obey the user defined tolerance δ it is of course necessary to chose the size of each voxel ε smaller than δ. Taking into account that the diagonal of a voxel is 2ε, the two layers for $\mathcal{V}_2$ as well as another layer for the conservative voxelization of $\mathcal{V}_0$ we have to chose $\varepsilon < \delta/3\sqrt{(3)}$. That is, for a reasonable scenario with an area of interest of about $1m^3$ and a user defined tolerance of $1mm$ this would result in about $5200^3 \simeq 140 * 10^9$ voxel, which brings memory consumption onto the table. In order to decrease the memory usage we store a voxelization as an octree which we, similar to [5], internally represented by a hash set. A node is marked as occupied by its pure existence in the hash set. In case all 8 children of a node are marked the node is marked and the children can be deleted. A voxel is not covered if its leaf and non of its ancestors exists in the hash set. Testing containment as well as insertion is in $O(\log 1/\varepsilon)$ However, the most important property is that for a reasonable $\mathcal{SV}$ one can expect a memory consumption that is proportional to $\partial\mathcal{SV}$, i.e., quadratic in $1/\varepsilon$.

Assuming a linear interpolation every triangle gives rise to a deformed prism between two trajectory positions, see also Figure 2. Thus inserting all voxels of a conservative voxelization of each such prism would directly yield $\mathcal{V}_0$. Since generating all these voxels is to expensive we follow a culling heuristic by [3] that tries to rule out those triangles (of the prisms) that do

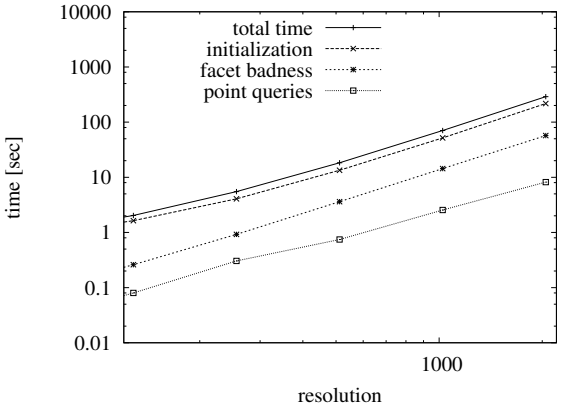


Figure 3: Total time in relation to main functions.

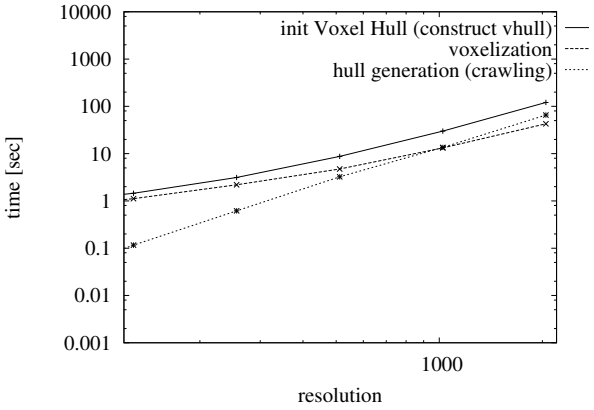


Figure 4: Times for initialization phase.

not contribute to the boundary. This is based on local criteria of the mesh in relation to the current direction of the motion. Only the remaining triangles are then voxelized. The voxelization is based on a simple subdivision scheme. Starting from the initial cube, the boxes that still intersect the triangle are subdivided until the required resolution is reached. The intersection test is based on the separating axes theorem. The result $\partial\tilde{\mathcal{V}}_0$ is a subset of $\mathcal{V}_0$ and contains $\partial\mathcal{V}_0$. In a next step we generate $\partial\mathcal{V}_1 = \mathcal{V}_1 \setminus \mathcal{V}_0$ by crawling along the outer boundary of $\partial\tilde{\mathcal{V}}_0$. We then throw away $\partial\tilde{\mathcal{V}}_0$ and generate $\mathcal{V}_0$ by filling $\partial\mathcal{V}_1$. $\mathcal{V}_1$ is then generated by simply adding $\partial\mathcal{V}_1$ to a copy of $\mathcal{V}_0$. $\mathcal{V}_2$ is created in a similar fashion. Since $\partial\tilde{\mathcal{V}}_0$ and $\partial\mathcal{V}_1$ are stored in a plain hash set the crawling is, for reasonable volumes, in $O((1/\varepsilon)^2)$. Note that the filling is not cubic since it uses an hierarchic scheme, which we can no discuss here due to limited space.

4 Preliminary Results

All benchmarks were measured on a Intel(R) Core(TM) i5 CPU M 450 with 2.40GHz with 512 kB cache under Ubuntu Linux and the GNU C++ compiler v4.4 with optimizations (`-O2`). However, we only use one CPU since we did not parallelize any part of the approach yet.

The swept Stanford Bunny, Figures 1 and 5, were generated from an initial mesh with 8100 triangles and trajectories of size 12/50. The resolution was set to $2^{10} = 1024$ which corresponds to $\delta \simeq 0.005$. One can see that the mesh nicely adapts to the local complexity of $\mathcal{SV}$, with sparse areas in well behaved regions. The resulting mesh has only about 20k/10k boundary facets. Figure 3 shows the obtained times for the Stanford Bunny shown in Figure 1. Note that times and resolution are given in a logarithmic scale. The graphs show the total time in relation to three major components: the time for the initialization phase, time spend in point classification and facet classification. The sum of the later two is essentially the time spend in the mesh generation phase. Thus already for such a small example the initialization phase clearly dominates the time used overall.

It turns out that most of the time of the initialization phase is currently spend in the voxelization of all triangles that where not culled and the computation of the hull surrounding it. All other steps such as the hierarchical filling of $\mathcal{V}_0$ and the offset computation to compute $\mathcal{V}_1$ and $\mathcal{V}_2$ from $\mathcal{V}_0$ are not relevant. Figure 4, shows a more detailed plot of the time spend in the hull computation. Initially, the voxelization is clearly the dominating part, but for high resolution the hull computation eventually takes over. This is due to the fact that the offset computation in this part is more expensive since the volume is not filled and thus the size of the octree is much larger. However, for larger models and larger trajectories the first part would currently be the most dominating time factor.

5 Further Work

Though our results are already very promising we already see several, partially obvious, optimizations that where not yet applied due to lack of time.

Using only one offset would significantly improve memory usage but would also improve ε to $\delta/4$. It only requires a slightly modified point generation function. On the other hand, the new scheme would only leave a corridor of width about $\delta/2$ to place triangles. This would probably have a negative impact on the size of the resulting mesh. For the current scheme the width is about $2\delta/3$.

It is clear that the voxel generation can be easily parallelized. For instance, each CPU may be dedicated to a certain volume in space. The resulting octrees may be merged afterwards. That is, it is easy to parallelize the currently most time consuming part of the algorithm. An obvious alternative is to move

the voxel generation to the GPU.

So far our implementation can not handle input meshes that are not manifolds. However, this is just due to missing case distinction in the current code handling the culling. In principal the approach is able to easly handle meshes that are not watertight or have other topological inconsistencies. For instance, the voxelization would fill the gap if it is smaller than ε. Note that this also means that the approach ignores narrow tunnels to maybe rather large caves inside the model. In case the tunnel has diameter around delta the hull computation explores the cave, but it may appear as a closed void since the tunnel is closed due to the subsequent offset computations. Note that this does not contradict our guarantees since we are only interested in the one-sided Hausdorff-distance.

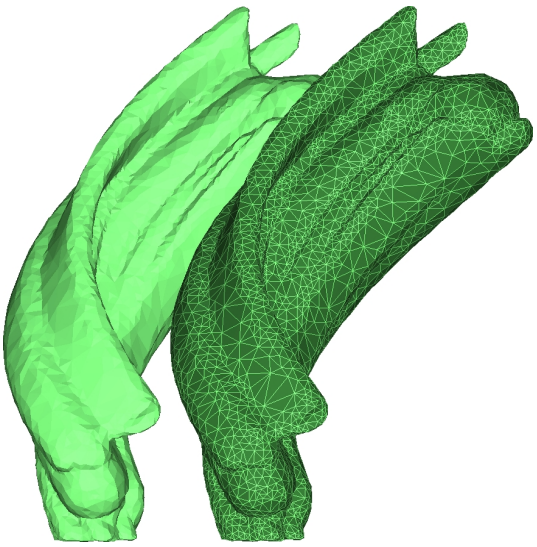


Figure 5: Bunny swept along a trajectory consisting of 50 transformations.

6 Conclusion

To the best of our knowledge this is the first conservative approache for swept volume mesh genearation that also guarantees an approximation quality in terms of the one-sided Hausdorff distance. The resultings meshes are of high quality and very reasonable size, which makes them ideal for further processing in industrial applications. Moreover, our prelimiary benchmark results indicate that the approach should even applicalbe for very large inputs, in particular, once the parallization of the initialization phase is in place.

For recent improvements, more examples and other supplementary material we refer to our website: `http://acg.cs.tau.ac.il/projects/internal-projects/swept-volume/project-page`.

Acknowledgments

We thank R. Erbes for supplying us with the voxelization algorithm for triangles.

References

[1] K. Abdel-Malek, D. Blackmore, and K. Joy. Swept volumes: Foundations, perspectives, and applications. *International Journal of Shape Modeling*, 23(5):1–25, 2004.

[2] S. Abrams and P. K. Allen. Computing swept volumes. *Journal of Visualization and Computer Animation*, 11:69–82, 2000.

[3] M. Campen and L. Kobbelt. Polygonal boundary evaluation of minkowski sums and swept volumes. In *Eurographics Symposium on Geometry Processing (SGP 2010)*, 2010.

[4] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

[5] S. F. Frisken and R. N. Perry. Simple and efficient traversal methods for quadtrees and octrees. *Journal of Graphics Tools*, 7(3):1–12, 2002.

[6] J. C. Himmelstein, E. Ferre, and J.-P. Laumond. Swept volume approximation of polygon soups. In *ICRA*, pages 4854–4860, 2007.

[7] Y. J. Kim, G. Varadhan, M. C. Lin, and D. Manocha. Fast swept volume approximation of complex polyhedral models. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 11–22, New York, NY, USA, 2003. ACM.

[8] L. Rineau and M. Yvinec. *3D Surface Mesher*, 3.2 edition, 2006. CGAL User and Reference Manual.

[9] A. von Dziegielewski, R. Erbes, and E. Schömer. Conservative swept volume boundary approximation. In *SPM '10: Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, pages 171–176, New York, NY, USA, 2010. ACM.

[10] S. Zelinka and M. Garland. Permission grids: Practical, error-bounded simplification. *ACM Transactions on Graphics*, 21:2002, 2002.

[11] X. Zhang, Y. J. Kim, and D. Manocha. Reliable sweeps. In *SPM '09: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pages 373–378, New York, NY, USA, 2009. ACM.

Switch-regular Upward Planar Drawings with Low-degree Faces*

Walter Didimo†

Abstract

Upward planar drawings of digraphs are crossing free drawings where the vertices are mapped to points of the plane and all edges are simple curves that flow in the upward direction, according to their orientation. *Switch-regular upward planar drawings* are upward planar drawings with additional properties. They have both theoretical and practical interest in graph drawing. The complexity of deciding whether an embedded planar digraph admits a switch-regular upward planar drawing is still unknown. A linear time testing algorithm is known for digraphs whose underlying graph is a tree. In this paper we show that the switch-regular upward planarity testing problem can be solved in polynomial time also for planar embedded digraphs with faces of degree at most six.

1 Introduction

Let G be a planar digraph. An *upward planar drawing* of G is a planar drawing such that each vertex of G is mapped to a distinct point of the plane and all the edges of G are drawn as Jordan arcs monotonically increasing in the upward direction, according to their orientation. Not all planar digraphs admit an upward planar drawing, and the problem of deciding whether a planar digraph admits an upward planar drawing has been widely investigated in the literature. This problem is referred to as the *upward planarity testing problem*. In the variable embedding setting Garg and Tamassia showed that the upward planarity testing problem is NP-complete [12]; consequently, branch-and-bound algorithms and fixed-parameter tractable algorithms have been described [2, 7, 13]. Also, it has been shown that the problem can be solved in polynomial time for outerplanar digraphs [15], series-parallel digraphs [11], and single-source digraphs [4, 14]. In the fixed embedding setting, Bertolazzi *et al.* showed that the upward planarity testing can be executed in polynomial time by reducing it to a network flow problem [3].

The topological and combinatorial properties of upward planar drawings have also been investigated. In particular, Di Battista and Liotta discovered and characterized an interesting sub-family of upward planar drawings whose embedding has some interesting properties of "regularity" [9]. Roughly speaking, the *upward embedding* of an upward planar drawing describes the circular sequence of "small" angles (i.e., angles smaller than π) and "large" angles (i.e., angles larger than π) at the *switch vertices* of each face; a vertex v of a face f is a switch-vertex if its incident edges in f are both leaving or both entering v. The upward embedding of an upward planar drawing is said to be *switch-regular* if: (i) the boundary of each internal face contains at most one maximal subsequence of small angles of length greater than one; (ii) the external boundary does not contain two consecutive small angles. Figure 1 shows two different upward planar drawings of the same embedded planar digraph: The first drawing has a switch-regular upward embedding, while the second drawing has a non-switch regular upward embedding. An upward planar drawing with a switch-regular upward embedding is also called a *switch-regular upward planar drawing*.

Finding switch-regular upward planar drawings has practical relevance for the design of efficient graph drawing checkers [9] and of effective compaction algorithms [10]. The complexity of deciding whether a planar digraph admits a switch-regular upward planar drawing is still unknown, even in the fixed embedding setting. Binucci *et al.* proved that this problem can be solved in linear time in the variable embedding setting for digraphs whose underlying graph is a tree [5, 6].

In this paper we study the switch-regular upward planarity testing problem in the fixed embedding setting. We prove that, given an embedded planar digraph G with n vertices and face-degree at most six, it is possible to decide in $O(n^2)$ time whether G admits a switch-regular upward planar drawing, and to compute one in the positive case.

The remainder of the paper is structured as follows. In Section 2 we recall some basic definitions and results about upward planarity and switch-regular embeddings. In Section 3 we present our testing and embedding algorithm. Some open problems are listed in Section 4.

2 Basic definitions and results

We assume familiarity with the basic concepts of graph drawing and graph planarity [8]. Let G be an embedded planar digraph. A vertex v of G is *bimodal*

*The research behind this work started at the first Bertinoro Workshop on Graph Drawing, in 2006.

†Dip. di Ing. Elettr. e dell'Informaz., Università degli Studi di Perugia didimo@diei.unipg.it

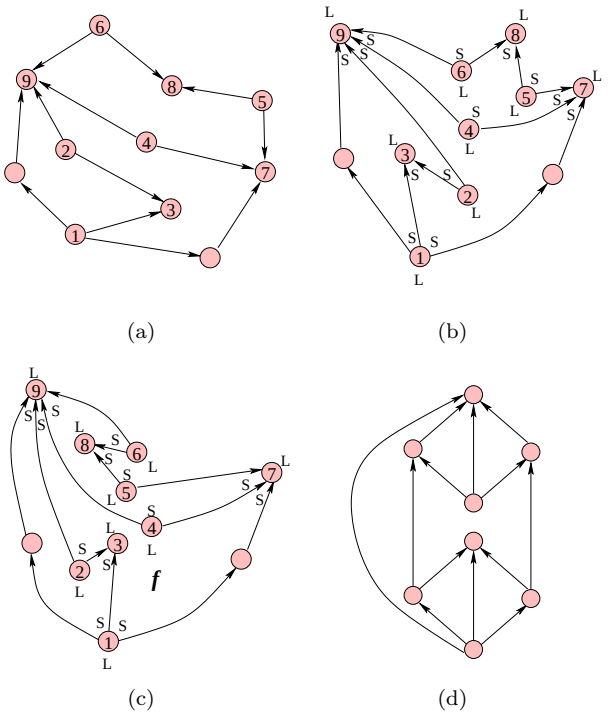


Figure 1: (a) A planar embedded digraph G; source and sink vertices are labeled with integer numbers. (b) An upward planar drawing of G whose upward embedding is switch-regular; S and L labels denote small and large angles, respectively; (c) An upward planar drawing of G whose upward embedding is not switch-regular (the external face and the internal face f are not switch-regular). (d) A planar digraph that does not admit a switch-regular upward planar drawing (the internal face of degree six is not switch-regular in any upward planar drawing of the digraph).

if all incoming edges of v (and hence all outgoing edges of v) are consecutive in the circular clockwise order around v. G is called *bimodal* if all its vertices are bimodal. Acyclicity and bimodality are necessary, but not sufficient, conditions for the upward planar drawability of an embedded planar digraph [3].

Let f be a face of G and suppose that the boundary of f is traversed counterclockwise. If $e_1 = (u_1, v)$ and $e_2 = (v, u_2)$ are two edges encountered in this order along the boundary of f, the triplet $s = (e_1, v, e_2)$ is called an *angle of* f. Observe that, e_1 and e_2 may coincide if G is not biconnected. Angle s is called a *switch of* f if e_1 and e_2 are both incoming edges or both outgoing edges of v: In the first case s is also called a *sink-switch of* f, while in the second case it is a *source-switch of* f. It is immediate to see that the number of source-switches of f equals the number of sink-switches of f. We denote by $2n_f$ the total number of switches of f. The *capacity of* f is denoted by c_f and it is defined by $c_f = n_f - 1$ if f is an internal face and by $c_f = n_f + 1$ if f is the external face. For a bimodal digraph G, an assignment of its sources and sinks to the faces of G is called an *upward consistent assignment* if: (i) Each source and each sink is assigned to exactly one of its incident faces; (ii) for each face f exactly c_f sources and sinks on the boundary of f are assigned to f. The following theorem characterizes the class of embedded planar digraphs that admit an upward planar drawing.

Theorem 1 [3] *An acyclic embedded planar bimodal digraph is upward planar if and only if it admits an upward consistent assignment.*

The *upward embedding* corresponding to an upward consistent assignment of G is a planar embedding of G with labels at the switches of every face. Namely, a switch $s = (e_1, v, e_2)$ of f is labeled L when v is a source or a sink assigned to f, while s is labeled S otherwise. If f is a face of an upward embedding, the circular list of labels of f is denoted by σ_f. Also, S_{σ_f} and L_{σ_f} denote the number of S and L labels of f, respectively. The following property holds.

Property 1 [3] *If f is a face of an upward embedding then $S_{\sigma_f} = L_{\sigma_f} + 2$ if f is internal, and $S_{\sigma_f} = L_{\sigma_f} - 2$ if f is external.*

An upward planar drawing of a digraph G can be constructed for a given upward embedding of G; this drawing is such that each switch labeled L forms a geometric angle larger than π, while each switch labeled S forms a geometric angle smaller than π.

An internal face f of an upward embedding is called *switch-regular* if σ_f does not contain two distinct maximal subsequences σ_1 and σ_2 of S labels such that $S_{\sigma_1} > 1$ and $S_{\sigma_2} > 1$. The external face f is *switch-regular* if σ_f does not contain two consecutive S labels. An upward embedding is *switch-regular* if all its faces are switch-regular. We say that a digraph G is *switch-regular* if it admits a switch-regular upward embedding.

Figure 1 shows examples of switch-regular and non-switch regular upward embeddings for the same digraph G. Both the upward embeddings preserves the planar embedding of G. Figure 1(d) shows a non-switch regular digraph.

3 Digraphs with face-degree at most six

Here, we present a polynomial-time testing algorithm for embedded planar digraphs whose faces have at most six switches. Clearly, this condition is always true if the faces have degree at most six.

We reduce the switch-regular upward planarity testing problem on G to the computation of an integer feasible flow in a suitable network $N(G)$ constructed from G. Our flow model is inspired by that proposed by Bertolazzi *et al.* for the classical upward

planarity testing problem [3]. However, our network is enhanced with extra elements (nodes and arcs), in order to guarantee that every face is assigned a switch-regular embedding when possible.

The flow network $N'(G)$ of Bertolazzi *et al.* is based on Theorem 1. It consists of a node $n(u)$ for each source or sink vertex u of G and of a node $n(f)$ for each face f of G. Vertices $n(u)$ and $n(f)$ are called *vertex-nodes* and *face-nodes*, respectively. Also $N'(G)$ has a directed arc $a(u, f) = (n(u), n(f))$ if u belongs to f; the arc has upper capacity one and lower capacity zero. A unit of flow on an arc $a(u, f)$ represents a large angle formed at u inside f (i.e., a label L at u inside f). Each vertex-node $n(u)$ supplies flow one, which represents the fact that u must form exactly one large angle inside one of its incident faces. Each face-node $n(f)$ demands an amount of flow equal to the capacity c_f of face f, which represents the fact that f must arrange exactly c_f large angles. Hence, there exists an upward consistent assignment describing an upward embedding of G if and only if $N(G')$ admits an integer feasible flow.

We construct a flow network $N(G)$ by enhancing $N'(G)$. First we want to restrict all possible upward consistent assignments to those assignments whose corresponding upward embeddings have switch-regular internal faces only. If f is an internal face of G, we distinguish between two possible cases:

f **has either 2 or 4 switch-angles.** In this case c_f is either 0 or 1, and f has to arrange at most one large angle. Hence, f is switch-regular in every upward embedding of G. No local modification of the network is required for this case.

f **has 6 switch-angles.** In this case $c_f = 2$ and then f has to arrange two large angles. There are three forbidden configurations, shown in Figure 2, that we have to avoid. To this aim, we locally modify the network as follows (see also Figure 2(d)): If u and v are sources or sinks incident to f, such that there are two switch-angles along the boundary of f between u and v (in both the two circular directions), then we remove arcs $a(u, f)$ and $a(v, f)$ and add the following extra elements: (*i*) A node $n_{uv}(f)$ that neither supplies nor demands flow (it is a transient node); (*ii*) Two directed arcs $(u, n_{uv}(f))$ and $(v, n_{uv}(f))$, each having upper capacity one and lower capacity zero. (*iii*) A directed arc $(n_{uv}(f), n(f))$, with upper capacity one and lower capacity zero.

Since $n_{uv}(f)$ is a transient node and the upper capacity of arc $(n_{uv}(f), n(f))$ is one, such a local modification of the network avoids that both u and v receive an L label inside f, but leaves the possibility that one of them has such a label.

For the external face h of G, we must avoid those upward embeddings having two consecutive S labels along the boundary of h. If h has less than six switches, then h is switch-regular in any upward embedding of G. If h has six switches, then $c_h = 4$ and h has to arrange four L labels and two S labels. Hence, there are at most 15 possible arrangements of S and L labels for the switches of h, 9 of which correspond to switch-regular upward embeddings of h. For each of these switch-regular upward embeddings, we can locally fix in $N(G)$ the flow associated with that upward embedding, and look for an integer feasible flow within this constraint. To fix a flow of value x on an arc of $N(G)$, it is sufficient to set upper and lower capacity of that arc equal to x.

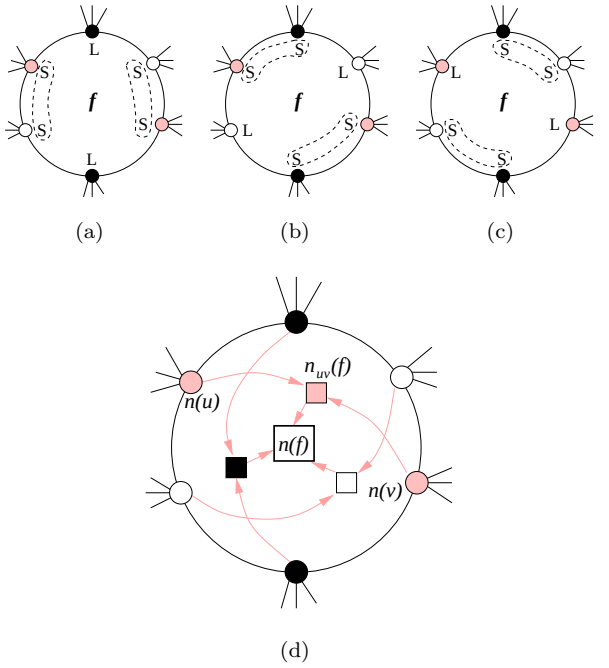


Figure 2: (a)–(c) The three forbidden upward embeddings for an internal face with six switches. Maximal subsequences of S labels are bounded by dashed closed curves. (d) Flow network for an internal face with six switches. All arcs have lower capacity zero and upper capacity one.

Figure 3 shows the flow network for the embedded digraph of Figure 1(a) and a feasible flow corresponding to the switch-regular upward embedding of Figure 1(b).

Theorem 2 *Let G be an embedded planar digraph with n vertices, such that every face has at most six switches. There exists an $O(n^2)$ time algorithm that tests whether G admits a switch-regular upward planar drawing and that computes such a drawing in the positive case.*

Proof. Construct the flow network $N(G)$ from the planar embedding of G. This is done in $O(n)$ time.

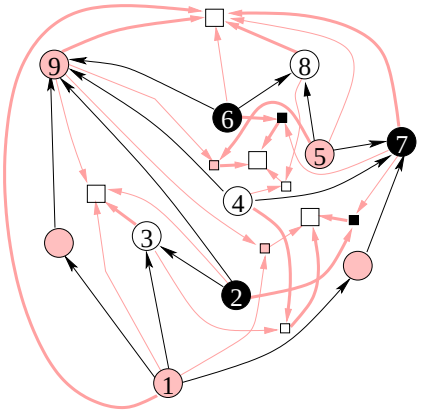


Figure 3: The flow network for the embedded digraph of Figure 1(a) (grey arcs). The flow in the network corresponds to the switch-regular embedding of Figure 1(b). Thick arcs have flow one, while the others have flow zero.

For each possible switch-regular upward embedding of the external face h of G, fix the corresponding flow on $N(G)$ and look for an integer feasible flow in $N(G)$. If at least one feasible flow is found, then the test is positive, otherwise it is negative. Since both the size of $N(G)$ and the value of the flow are $O(n)$, an integer feasible flow in $N(G)$ can be computed in $O(n^2)$ time with a standard algorithm [1]. Also, we need to run such an algorithm a constant number of times, because there are at most 9 switch-regular upward embeddings for h. Hence, it is possible to test the existence of an integer feasible flow corresponding to a switch-regular upward embedding of G in $O(n^2)$ time. If such a flow exists, the label assignment associated with its values describes a switch-regular upward embedding of G, and an upward planar drawing with this upward embedding is computed with the same linear time procedure described in [3]. □

Corollary 1 *Let G be an embedded planar digraph with n vertices and face-degree at most six. There exists an $O(n^2)$ time algorithm that tests whether G admits a switch-regular upward planar drawing and that computes such a drawing in the positive case.*

4 Open Problems

The main open question is what is the complexity of the switch-regular upward planarity testing problem in the general case. It is also interesting to extend the approach presented in this paper to design a fixed parameter tractable algorithm whose time complexity depends on the maximum face degree.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. North-Holland, 1990.

[2] P. Bertolazzi, G. Di Battista, and W. Didimo. Quasi-upward planarity. *Algorithmica*, 32(3):474–506, 2002.

[3] P. Bertolazzi, G. Di Battista, G. Liotta, and C. Mannino. Upward drawings of triconnected digraphs. *Algorithmica*, 6(12):476–497, 1994.

[4] P. Bertolazzi, G. Di Battista, C. Mannino, and R. Tamassia. Optimal upward planarity testing of single-source digraphs. *SIAM Journal on Computing*, 27:132–169, 1998.

[5] C. Binucci, E. D. Giacomo, W. Didimo, and A. T. Rextin. Switch-regular upward planar embeddings of trees. In *WALCOM 2010*, volume 5942 of *Lecture Notes Comput. Sci.*, pages 58–69, 2010.

[6] C. Binucci, E. D. Giacomo, W. Didimo, and A. T. Rextin. Switch-regular upward planarity testing problem of directed trees. *Journal of Graph Algorithms and Applications*, to apper.

[7] H. Chan. A parameterized algorithm for upward planarity testing. In *Proc. ESA '04*, volume 3221 of *LNCS*, pages 157–168, 2004.

[8] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[9] G. Di Battista and G. Liotta. Upward planarity checking: "faces are more than polygon". In *Graph Drawing (Proc. GD '98)*, volume 1547 of *LNCS*, pages 72–86, 1998.

[10] W. Didimo. Upward planar drawings and switch-regularity heuristics. *Journal of Graph Algorithms and Applications*, 10(2):259–285, 2006.

[11] W. Didimo, F. Giordano, and G. Liotta. Upward spirality and upward planarity testing. *SIAM Journal on Discrete Mathematics*, 23(4):1842–1899, 2009.

[12] A. Garg and R. Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM Journal on Computing*, 31(2):601–625, 2001.

[13] P. Healy and K. Lynch. Fixed-parameter tractable algorithms for testing upward planarity. *International Journal of Foundations of Computer Science*, 17(5):1095–1114, 2006.

[14] M. D. Hutton and A. Lubiw. Upward planarity testing of single-source acyclic digraphs. *SIAM Journal on Computing*, 25(2):291–311, 1996.

[15] A. Papakostas. Upward planarity testing of outerplanar dags. In *Proc. GD'94*, volume 894 of *LNCS*, pages 298–306, 1995.

Implicit Flow Routing on Triangulated Terrains

Mark de Berg Herman Haverkort Constantinos P. Tsirogiannis*

Abstract

Flow-related structures on terrains are defined in terms of paths of steepest descent (or ascent). A steepest descent path on a polyhedral terrain $\mathcal{T}$ with n vertices can have $\Theta(n^2)$ complexity, since at worst case the path can cross $\Theta(n)$ triangles for $\Theta(n)$ times each. We present a technique for tracing a path of steepest descent on $\mathcal{T}$ in $O(n \log n)$ time implicitly, without computing all the intersection points of the path with the terrain triangles.

1 Introduction

Background and motivation. In many applications it is necessary to visualize, compute, or analyze flows on a height function defined over some 2- or higher-dimensional domain. Often the direction of flow is given by the gradient and the domain is a region in $\mathbb{R}^2$. The flow of water in mountainous regions is a typical example of this. Modeling and analyzing water flow is important for predicting floods, planning dams, and other water-management issues. Hence, flow modeling and analysis has received ample attention in the GIS community [6, 7, 8, 9].

In GIS, mountainous regions are usually modeled as a DEM or as a TIN. A DEM (digital elevation model) is a uniform grid, where each grid cell is assigned an elevation. Because of the discrete nature of DEMs, it is hard to model flow in a natural and accurate way. A TIN (triangulated irregular network) is obtained by assigning elevations to the vertices of a two-dimensional triangulation; it is the model we adopt in this paper. In computational geometry, a TIN is usually referred to as a *(polyhedral) terrain*. One advantage of polyhedral terrains over DEMs is that one can use a non-uniform resolution, using small triangles in rugged areas and larger triangles in flat areas. Another advantage is that the surface defined by a polyhedral terrain is continuous, which makes flow modeling more natural. Indeed, the standard flow model on polyhedral terrains is simply that water follows the direction of steepest descent. To make the flow direction well defined, it is then often assumed—and we will also make this assumption—that the direction of steepest descent is unique for every point on the terrain. For instance, the terrain should not contain horizontal triangles.[1]

There are several important structures related to the flow of water on a polyhedral terrain $\mathcal{T}$. The simplest structure is the path that water would follow from a given point p on the terrain. This path is called the *trickle path* and, as already mentioned, in our model it is simply the path of steepest descent. Another important structure is the *watershed* of a point p on $\mathcal{T}$, which is the set of all points on $\mathcal{T}$ from which water flows to p. In other words, it is the set of points whose trickle path contains p.

Unfortunately, the combinatorial complexity of these structures can be quite high. For instance, De Berg *et al.* [3] showed that there are terrains of n triangles on which certain trickle paths cross $\Theta(n)$ triangles each $\Theta(n)$ times, resulting in a path of complexity $\Theta(n^2)$. McAllister [1] and McAllister and Snoeyink [2] showed that the total complexity of the watershed boundaries of all local minima can be $\Theta(n^3)$. This is due to the fact that at worst case the boundary of a watershed can consist of $\Theta(n)$ paths of steepest gradient, each of $\Theta(n^2)$ complexity.

For *fat terrains*, where the angles of the terrain triangles are lower-bounded by a constant, the situation is somewhat better: here the worst-case complexity of a single path of steepest ascent/descent is $\Theta(n)$ [4]. The complexity of a watershed, however, can still be $\Theta(n^2)$.

It is not always necessary, however, to explicitly compute the structure of interest. For example, it may be sufficient to compute only the point where a path of steepest descent ends, rather than all the intersections points of the path with the terrain triangles. Is it possible thus to compute for a point p on $\mathcal{T}$ the point where the trickle path from p ends without explicitly computing the path itself, thereby avoiding a worst-case running time of $\Theta(n^2)$?

Our results. Inspired by the above, we study the problem of implicitly tracing paths of steepest descent or ascent on a polyhedral terrain $\mathcal{T}$ with n vertices. we give an $O(n \log n)$ algorithm that finds out where the trickle path of a given point p ends, without con-

*Dept. of Mathematics and Computer Science, Eindhoven University of Technology, mdberg@win.tue.nl, cs.herman@haverkort.net, ctsirogi@win.tue.nl

[1] This can of course be ensured by a small perturbation of the elevations of the terrain vertices, but even small perturbations may have undesirable effects on the water flow. How to deal with horizontal triangles is therefore an important research topic in itself.

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

structing the actual path (which would take $\Theta(n^2)$ time in the worst case). Our algorithm can also report all the triangles crossed by the path in the same amount of time.

Terminology and notation. For a terrain $\mathcal{T}$ we denote the set of its edges by E, and the set of its vertices by V. Edges in E are defined to be open, that is, they do not include their endpoints. For any point p we denote its z-coordinate by $z(p)$. For an edge $e \in E$ incident to a triangle t we call e an *out-edge* of t if e receives water from the interior of t through the direction of steepest descent. Otherwise we call e an *in-edge* of t. We call e a *valley* edge if e is an out-edge for both of its incident triangles, we call e a *transfluent* edge if e is an out-edge for only one incident triangle, and we call e a *ridge* edge if it is an in-edge for both of its incident triangles.

2 Computing the triangles crossed by a trickle path

Let $\mathcal{T}$ be a terrain with n triangles, and let p be the point for which we want to compute the point where *trickle*(p) ends. As we only want to find where *trickle*(p) ends, we do not want to explicitly compute all intersection points between *trickle*(p) and the terrain edges. To avoid this, each time we encounter a sequence of edges that we crossed before, we jump to the first edge that we have not encountered so far. We can detect features that we already crossed, because we *mark* them the first time we hit them. Next we show how to do the above.

Define an *EV-sequence* to be the (ordered) sequence of terrain edges and vertices crossed by some path on $\mathcal{T}$. For a point $q \in$ *trickle*(p), let $\mathcal{S}(q)$ denote the EV-sequence crossed by the part of *trickle*(p) from p to q. Consider a point $q \in$ *trickle*(p) and let $\mathcal{S}(q) = f_1 f_2 \cdots f_k$. Let j be the largest index such that the feature f_j occurs at least twice in $\mathcal{S}(q)$, and let i be the largest index with $i < j$ such that $f_i = f_j$. We call $f_i f_{i+1} \cdots f_j$ the *last cycle* of $\mathcal{S}(q)$, and we call $f_{j+1} \cdots f_k$ the *last chain* of $\mathcal{S}(q)$; see Fig. 1(i). We need the following lemma.

Lemma 1 *Let f be a feature in $\mathcal{S}(q)$ that only occurs before the last cycle of $\mathcal{S}(q)$. Then trickle(q) cannot cross f.*

Proof. Let $\mathcal{S}(q) = f_1, \ldots, f_k$ and let $f_i, \ldots, f_j$ be the last cycle of $\mathcal{S}(q)$. Let $e = f_i = f_j$ and let r_i and r_j be the intersection points of *trickle*(p) with e that correspond to f_i and f_j, respectively. Let $\pi(p, r_i)$ be the part of *trickle*(p) from p to r_i and let $\pi(r_i, r_j)$ be the part of *trickle*(p) between r_i and r_j. Note that *trickle*$(q) \subset$ *trickle*(r_j). Define $P := \pi(r_i, r_j) \cup \overline{r_i r_j}$. Then P is the boundary of a simple polygon—see Fig.1(i), where this polygon is depicted grey. Since trickle-paths cannot self-intersect and e can be crossed in only one direction by a trickle path, one of the paths $\pi(p, r_i)$ and *trickle*(r_j) lies completely inside P while the other lies completely outside P. This implies that a feature intersecting $\pi(p, r_i)$ can only intersect *trickle*(q) if that feature intersects $\pi(r_i, r_j)$ and, hence, occurs in the last cycle.

□

Now imagine tracing *trickle*(p) and suppose we reach an edge e that we already crossed before. Let q be the point on which *trickle*(p) crosses e this time. After crossing e again, we may cross many more edges that we already encountered. Our goal is to skip these edges and immediately jump to the next new edge on the trickle path. By Lemma 1, the already crossed edges are either in the last cycle or in the last chain of $S(q)$. In fact, since q lies on an already crossed edge, the last chain is empty and so the edges we need to skip are all in the last cycle. Thus we store the last cycle in a data structure T_{cycle}—we call this structure the *cycle tree*—that allows us to jump to the next new edge by performing a query *FindExit*(T_{cycle}, q). More precisely, if $\mathcal{C} = f_i, \ldots, f_k$ denotes the cycle stored in T_{cycle} and q is a point on f_i, then *FindExit*(T_{cycle}, q) reports a pair $(f_{\text{exit}}, q_{\text{exit}})$ such that f_{exit} is the first feature crossed by *trickle*(q) that is not one of the features in $\mathcal{C}$ and q_{exit} is the point where *trickle*(q) hits f_{exit}. The cycle tree stores the last cycle encountered so far in the trickle path, thus we have to update this tree according to the changes in the last cycle.

Besides the cycle tree we also maintain a list L which stores the last chain of $\mathcal{S}(q)$; these edges may have to be inserted into T_{cycle} later on. This leads to the following algorithm.

Algorithm *ExpandTricklePath*$(\mathcal{T}, p)$
Input: A triangulated terrain $\mathcal{T}$ and a point p on the surface of $\mathcal{T}$.
Output: The point where *trickle*(p) ends and the edges crossed by this path.

1. Initialize an empty cycle tree T_{cycle} and an empty list L, and set $q := p$. If q lies on a feature f, then insert f into L.
2. **while** q is not a local minimum and flow from q does not exit the terrain
3. **do** ▷ Invariant: T_{cycle} stores the last cycle of $\mathcal{S}(q)$,
   ▷ and L stores its last chain.
4. Let f be the first feature that *trickle*(q) crosses after leaving from q, and let q' be the point where *trickle*(q) hits f.
5. $q := q'$
6. **if** f is not marked
7. **then** Mark f and append f to L.
8. **else** Update T_{cycle} and empty L.
9. Set $(f_{\text{exit}}, q_{\text{exit}}) :=$ *FindExit*(T_{cycle}, q), mark f_{exit}, and set $q := q_{\text{exit}}$.
10. Append f_{exit} to L (which is currently empty) and update T_{cycle}.
11. **return** q.

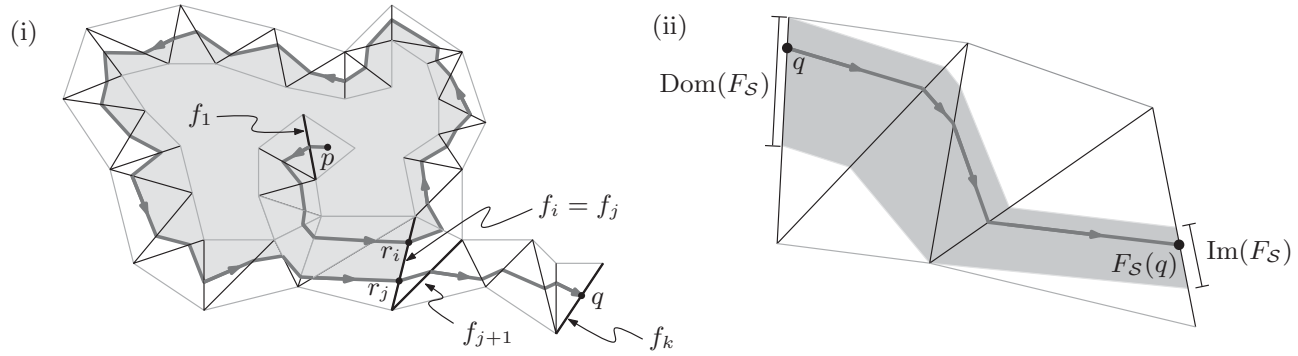


Figure 1: (i) The last cycle of the EV-sequence $\mathcal{S}(q)$ is $f_i, \ldots, f_j$, and the last chain is $f_{j+1}, \ldots, f_k$. (ii) The trickle function.

It is easy to see that the invariant holds after step 1 and that it is maintained correctly, assuming T_{cycle} is updated correctly in steps 8 and 10. This implies the correctness of the algorithm. Next we describe how to implement the cycle tree.

Consider an EV-sequence $\mathcal{S}$ without cycles and assume that there is some trickle path that crosses the features in $\mathcal{S}$ in the given order. Let $\text{first}(\mathcal{S})$ denote the first feature of $\mathcal{S}$ and let $\text{last}(\mathcal{S})$ denote its last feature. We define the *trickle function* $F_{\mathcal{S}} : \text{first}(\mathcal{S}) \to \text{last}(\mathcal{S})$ of the sequence $\mathcal{S}$ as follows. If the trickle path of a point $q \in \text{first}(\mathcal{S})$ follows the sequence $\mathcal{S}$ all the way up to $\text{last}(\mathcal{S})$, then $F_{\mathcal{S}}(q)$ is the point on $\text{last}(\mathcal{S})$ where *trickle*(q) hits $\text{last}(\mathcal{S})$. If, on the other hand, *trickle*(q) exits $\mathcal{S}$ before reaching $\text{last}(\mathcal{S})$, then $F_{\mathcal{S}}(q)$ is undefined. We denote the domain of $F_{\mathcal{S}}$ (the part of $\text{first}(\mathcal{S})$ where $F_{\mathcal{S}}$ is defined) by $\mathrm{Dom}(F_{\mathcal{S}})$, and we denote the image of $F_{\mathcal{S}}$ by $\mathrm{Im}(F_{\mathcal{S}})$. Since we assumed there is a trickle path crossing $\mathcal{S}$, both $\mathrm{Dom}(F_{\mathcal{S}})$ and $\mathrm{Im}(F_{\mathcal{S}})$ are non-empty. Fig. 1(ii) illustrates these definitions. Note that $\mathrm{Im}(F_{\mathcal{S}})$ is a single point when one of the features in $\mathcal{S}$ is a vertex. The following lemma follows from elementary geometry.

Lemma 2 *(i) The function $F_{\mathcal{S}}(q)$ is a linear function, and $\mathrm{Dom}(F_{\mathcal{S}})$ and $\mathrm{Im}(F_{\mathcal{S}})$ are intervals of $\text{first}(\mathcal{S})$ and $\text{last}(\mathcal{S})$, respectively. (ii) Suppose an EV-sequence $\mathcal{S}$ is the concatenation of EV-sequences $\mathcal{S}_1$ and $\mathcal{S}_2$. Then $F_{\mathcal{S}}$ can be computed from $F_{\mathcal{S}_1}$ and $F_{\mathcal{S}_2}$ in $O(1)$ time.*

Now consider an EV-sequence $\mathcal{S}(q) = f_1 \cdots f_k$ and let $\mathcal{C} = f_i, \ldots, f_j$ be the last cycle of $\mathcal{S}(q)$. The cycle tree T_{cycle} for $\mathcal{C}$ is a balanced binary tree, defined as follows.

- The leaves of T_{cycle} store the features $f_i, \ldots, f_{j-1}$ in order.
- For an internal node ν, let $lc[\nu]$ and $rc[\nu]$ denote its left and right child, respectively. Let $\mathcal{S}[\nu]$ denote the subsequence of $\mathcal{C}$ consisting of the features stored in the leaves below ν. Furthermore, let $\text{first}[\nu]$ and $\text{last}[\nu]$ denote the features stored in the leftmost and rightmost leaf below ν, respectively. Then ν stores the trickle function $F_{\mathcal{S}[\nu]}$, and the trickle function $F_{\mathcal{S}'[\nu]}$, where $\mathcal{S}'[\nu]$ is the sequence $f_\nu f'_\nu$ with $f_\nu = \text{last}[lc[\nu]]$ and $f'_\nu = \text{first}[rc[\nu]]$.

Lemma 3 *The function FindExit(T_{cycle}, q) can be implemented to run in $O(\log |\mathcal{C}|)$ time, where $|\mathcal{C}|$ is the length of the cycle stored in T_{cycle}.*

Proof. Imagine following *trickle*(q), starting at f_i, the first feature in $\mathcal{C}$. We will cross a number of features of $\mathcal{C}$, until we exit the cycle. (We must exit the cycle before returning to f_i again, because a trickle path cannot cross the same sequence twice without encountering another feature in between [3].) Let f^* be the feature of $\mathcal{C}$ that we cross just before exiting. We can find f^* in $O(\log |\mathcal{C}|)$ time by descending down T_{cycle} as follows.

Suppose we arrive at a node ν; initially ν is the root of T_{cycle}. We will maintain the invariant that f^* is stored in a leaf below ν. We will make sure that we have the point q_ν where *trickle*(q) crosses $\text{first}[\nu]$ available; initially $q_\nu = q$. When ν is a leaf we have found f^*, otherwise we have to decide in which subtree to recurse. The feature f^* is stored in the right subtree of an internal node ν if and only if

(i) $q_\nu \in \mathrm{Dom}(F_{\mathcal{S}[lc[\nu]]})$, which means *trickle*$(q_\nu)$ completely crosses $\mathcal{S}[lc[\nu]]$, and

(ii) $F_{\mathcal{S}[lc[\nu]]}(q_\nu) \in \mathrm{Dom}(F_{\mathcal{S}'[\nu]})$, meaning *trickle*$(q_\nu)$ reaches $\text{first}[rc[\nu]]$ after crossing $\mathcal{S}[lc[\nu]]$.

If these two conditions are met, we set $\nu := rc[\nu]$ and $q_\nu := F_{\mathcal{S}'[\nu]} \circ F_{\mathcal{S}[\nu]}(q_\nu)$, otherwise we set $\nu := lc[\nu]$.

Once we have found f^* and the point q^* where *trickle*(q) crosses f^*, we can compute the exit edge e_{exit} and point q_{exit} by inspecting the relevant triangle t incident to f^*: we just have to compute where the path of steepest descent from q^* exits t. □

It remains to explain how to update T_{cycle}. First consider step 8 of *ExpandTricklePath*. Suppose that,

just before q reaches f, we have $\mathcal{S}(q) = f_1 \cdots f_k$. Let $f_i \cdots f_j$ be the last cycle of $\mathcal{S}(q)$ (which is stored in T_{cycle}) and $f_{j+1} \cdots f_k$ its last chain (which is stored in L). We know that f has been crossed before. By Lemma 1 this implies $f = f_m$ for some $m \geqslant i$. We distinguish two cases.

- If $m > j$, then f occurs in the last chain and, hence, in L. Now after crossing f the last cycle becomes $f_m \cdots f_k f$. So updating T_{cycle} amounts to first emptying T_{cycle}, and then constructing a new cycle tree on $f_m \cdots f_k f$, which can be done by a bottom-up procedure in $O(|L|)$ time.
- If $i \leqslant m \leqslant j$ then f occurs in the last cycle. Then after crossing f the last cycle becomes $f_m \cdots f_j f_{j+1} \cdots f_k f$. (In the special case that $m = j$, we in fact have $f_i = f_j = f$ and the last cycle becomes $f_j f_{j+1} \cdots f_k f$.) We can now update T_{cycle} by deleting the features $f_1 \cdots f_{m-1}$, and inserting the features $f_{j+1} \cdots f_k$. (Recall that the last feature of a cycle is not stored in the cycle tree.) Inserting and deleting elements from an augmented balanced binary tree T_{cycle} can be done in logarithmic time in a standard manner.

Next consider the updating of T_{cycle} in step 10. Let $f_i \cdots f_j$ be the last cycle before step 9, where we jump to the first new feature crossed by the trickle path. Let f_m be the last feature we cross before we exit the cycle, that is, the feature f^* in the proof of Lemma 3. Then after the jump, the last cycle becomes $f_m \cdots f_{j-1} f_i \cdots f_m$. (Essentially, the cycle does not change, but its starting feature changes.) Thus, to update T_{cycle} we have to split T_{cycle} between f_{m-1} and f_m into two cycle trees T^1_{cycle} and T^2_{cycle}, then merge these cycles trees again but this time in the opposite order (that is, putting T^1_{cycle} to the right of T^2_{cycle} instead of to its left). Splitting and merging can be done in logarithmic time, if we use a suitable underlying tree such as a red-black tree. We obtain the following theorem.

Theorem 4 *Let $\mathcal{T}$ be a terrain with n triangles and let p a point on the surface of $\mathcal{T}$. Algorithm ExpandTricklePath$(\mathcal{T}, p)$ traces the trickle path of p in time $O(n \log C_{\max})$, where $C_{\max}$ is the length of the longest cycle in the EV-sequence of trickle(p).*

3 Applications to Other Drainage Structures

In the full version of this paper we show how we can use the presented method so as to derive an efficient mechanism that expands a collection of $\Theta(n)$ paths simultaneously. This mechanism combines the data structures that we describe above with a space-sweep algorithm. Based on this we derive $O(n \log n)$ time algorithms for:

- computing for each local minimum p of $\mathcal{T}$ the triangles contained in the watershed of p
- computing the surface network graph [5] of $\mathcal{T}$.

We have also designed an $O(n^2)$ time algorithm that computes the watershed area for each local minimum of $\mathcal{T}$.

References

[1] M. McAllister. A Watershed Algorithm for Triangulated Terrains. In *Proc. 11th Canadian Conference on Computational Geometry*, pages 103–106, 1999.

[2] M. McAllister and J. Snoeyink. Extracting Consistent Watersheds From Digital River And Elevation Data. *Annual Conference of the American Society for Photogrammetry and Remote Sensing*, 1999.

[3] M. de Berg, P. Bose, K. Dobrint, M. van Kreveld, M. Overmars, M. de Groot, T. Roos, J. Snoeyink and S. Yu. The Complexity of Rivers in Triangulated Terrains. In *Proc. 8th Canadian Conference on Computational Geometry*, pages 325–330, 1996.

[4] M. de Berg, O. Cheong, H. Haverkort, J. Lim and L. Toma. I/O-Efficient Flow Modeling on Fat Terrains. In *Proc. 10th Workshop on Algorithms and Data Structures*, pages 239–250, 2007.

[5] L. Čomić, L. De Floriani and L. Papaleo. Morse-Smale Decompositions for Modeling Terrain Knowledge. In *Proc. 7th International Conference on Spatial Information Theory*, pages 426–444, 2005.

[6] A. Frank, B. Palmer and V. Robinson. Formal Methods for the Accurate Definition of Some Fundamental Terms in Physical Geography. In *Proc. 2nd International Symposium Spatial Data Handling*, pages 585–599, 1986.

[7] S. Mackay and L. Band. Extraction and Representation of Nested Catchment Areas from Digital Elevation Models in Lake-Dominated Topography. *Water Resources Research Journal*, 34(4):897–901, 1998.

[8] O. Palacios-Velez and B. Cuevas-Renaud. Automated River-Course, Ridge and Basin Delineation from Digital Elevation Data. *Journal of Hydrology*, 86:299–314, 1986.

[9] D. Theobald and M. Goodchild. Artifacts of TIN-Based Surface Flow Modeling. In *Proc. of GIS/LIS'90*, pages 955–964, 1990.

[10] S. Yu, M. van Kreveld and J. Snoeyink. Drainage Queries in TINs: From local to global and back again. In *Proc. 7th International Symposium on Spatial Data Handling*, pages 13–1, 1996.

Right Angle Crossing Graphs and 1-planarity*

Peter Eades† Giuseppe Liotta‡

Abstract

A Right Angle Crossing Graph (also called RAC graph for short) is a graph that has a straight-line drawing where any two crossing edges are orthogonal to each other. A 1-planar graph is a graph that has a drawing where every edge is crossed at most once. We study the relationship between RAC graphs and 1-planar graphs in the extremal case that the RAC graphs have as many edges as possible. It is known that a maximally dense RAC graph with $n > 3$ vertices has $4n-10$ edges. We show that every maximally dense RAC graph is 1-planar. Also, we show that for every integer i such that $i \geq 0$, there exists a 1-planar graph with $n = 8+4i$ vertices and $4n-10$ edges that is not a RAC graph.

1 Introduction

Recent technological advances have generated torrents of relational data sets that are often represented and visually analyzed as graphs drawn in the plane. The large size of these data sets poses fascinating challenges to graph drawers: while a considerable portion of the existing graph drawing literature showcases elegant algorithms and sophisticated data structures under the assumption that the input graph is planar, most graphs are in fact non-planar in practice (see, e.g., [10, 11]).

In this context, it is worth recalling a few experimental studies that are motivating some of the current research directions about drawing non-planar graphs. Huang et al. [8, 9] prove that crossing edges significantly affect human understanding if they form acute angle, while crossing edges that form angles from about $\frac{\pi}{3}$ to $\frac{\pi}{2}$ guarantee good readability properties. Hence it makes sense to study drawings of graphs where such "sharp angle crossings" are forbidden. Purchase et al. [14, 15, 18]) prove that an edge is difficult to read if it is crossed by many other edges. Another research direction is therefore to study drawings of graphs where every edge can only be crossed a small number of times.

*Work supported in part by MIUR of Italy under project AlgoDEEP prot. 2008TFBWL4 and by an IVFR Grant of the Australian Government.

†School of Information Technologies, University of Sydney {peter}@it.usyd.edu.au

‡Dip. di Ing. Elettr. e dell'Informaz., Università degli Studi di Perugia {liotta}@diei.unipg.it

This paper studies the interplay between two families of non-planar drawings that fit into the above mentioned research directions.

More formally, a *drawing* of a graph G maps each vertex u of G to a distinct point p_u in the plane, each edge (u, v) of G a Jordan arc connecting p_u and p_v and not passing through any other vertex, and is such that any two edges have at most one point in common. A 1-*planar drawing* is a drawing of a graph where an edge can be crossed by at most another edge. A 1-*planar graph* is a graph that has a 1-planar drawing. A *straight-line drawing* is a drawing of a graph such that every edge is a straight-line segment. A *Right Angle Crossing drawing* (or *RAC drawing*, for short) is a straight-line drawing where any two crossing edges form right angles at their intersection point. A *Right Angle Crossing graph* (or *RAC graph*, for short) is a graph that has a RAC drawing.

Pach and Toth prove that 1-planar graphs with n vertices have at most $4n-8$ edges, which is a tight upper bound [13]. Korzhik and Mohar prove that recognizing 1-planar graphs is NP-hard [12]. Suzuki studies the combinatorial properties of the so-called *optimal* 1-*planar* graphs, i.e. those n-vertex 1-planar graph having $4n-8$ edges [16]. A limited list of additional papers on 1-planar graphs includes [4, 7]. Didimo et al. show that a RAC graph with $n > 3$ vertices has at most $4n-10$ edges and that this bound is tight [5]. Argyriou at al. prove that recognizing RAC graphs is NP-hard [2]. For recent references about RAC graphs and their variants see also [1, 3, 6, 17].

We focus on the relationship between RAC graphs and 1-planar graphs in the extremal case that the RAC graphs are as dense as possible. A RAC graph is *maximally dense* if it has $n > 3$ vertices and $4n-10$ edges. We prove the following.

Theorem 1 *Every maximally dense RAC graph is* 1-*planar. Also, for every integer i such that $i \geq 0$, there exists a* 1-*planar graph with $n = 8+4i$ vertices and $4n-10$ edges that is not a RAC graph.*

Section 2 considers a three coloring of the edges of a maximally dense RAC graph and studies the properties of the subgraphs induced by two of these three colors. Section 3 uses these properties to prove Theorem 1. For reasons of space, some proofs are sketched or omitted.

2 Red-blue-green Coloring of Maximally Dense RAC Graphs

Let G be a maximally dense RAC graph and let D be any RAC drawing of G. Let E be the set of the edges of D. In [5] the following 3-coloring of the edges of D (and hence of G) is described. Every edge is either a *red edge* or a *blue edge*, or a *green edge*. An edge is red if and only if it is not crossed by any other edge; a blue edge is only crossed by green edges, and a green edge is only crossed by blue edges. We call this 3-coloring of the edges of D a *red-blue-green coloring* of D and denote it as Π_{rbg}. Let $D_{rb} = (V, E_r \cup E_b)$ be the sub-drawing of D consisting of the red and blue edges and let G_{rb} be the corresponding subgraph of G. We call G_{rb} the *red-blue subgraph* of G induced by Π_{rbg} and we call D_{rb} the *red-blue sub-drawing* of D induced by Π_{rbg}. Note that, by construction, D_{rb} has no crossing edges and thus G_{rb} is a planar graph. We will always consider G_{rb} as a planar embedded graph, where the planar embedding is given by D_{rb}. Analogously we define the *red-green* subgraph of G induced by Π_{rbg}, denoted as G_{rg}, and the *red-green* sub-drawing of D induced by Π_{rbg}, denoted as D_{rg}. Also G_{rg} has the planar embedding of D_{rg}, and thus G_{rg} and G_{rb} have the same external face.

The next lemmas will particularly focus on the size and the coloring of some specific faces of the red-blue graph G_{rb}. We will consider its external face, denoted as f_{ext}, and its *fence faces*, defined as those internal faces that share at least one edge with f_{ext}. In the proofs that follow, we denote with m_r the number of red edges, with m_b the number of blue edges, and with m_g the number of green edges. Without loss of generality, we will assume from now on that our red-blue-green coloring is such that $m_b \geq m_g$. Also, we denote with f_{rb} the number of faces of G_{rb} and with n the number of its vertices.

Lemma 2 *[5] Every internal face of G_{rb} has at least two red edges. Also, all edges of f_{ext} are red.*

Lemma 3 *Face f_{ext} is a 3-cycle.*

Proof sketch. By Lemma 2, every internal face of G_{rb} has at least two red edges and all edges of f_{ext} are red. Hence, denoting with $|f_{ext}|$ the number of edges of f_{ext}, we have $m_r \geq (f_{rb} - 1) + \frac{|f_{ext}|}{2}$. Since G_{rb} is a planar graph, Euler's formula implies that $m_r + m_b \leq n + f_{rb} - 2$. It follows $m_b \leq n - 1 - \frac{|f_{ext}|}{2}$. Since also the red-green subgraph of G is planar and it has the same external face of G_{rb}, by Euler's formula we also have that $m_r + m_g \leq 3n - 3 - |f_{ext}|$. It follows that $m_r + m_b + m_g \leq 4n - 4 - \frac{3|f_{ext}|}{2}$. Observe that $|f_{ext}| \geq 5$ would imply $m_r + m_b + m_g < 4n - 10$, which is impossible because G is a maximally dense RAC graph. We now show that the external face of G_{rb} cannot be a 4-cycle either. By contradiction, assume that $|f_{ext}| = 4$. Consider first the case that some fence face of G_{rb} has more than 3 edges: Since $|f_{ext}| = 4$ and a fence face has size at least 4, we have $m_r + m_b \leq 3n - 8$. By the inequalities above, we also have $m_r \geq f_{rb} + 1$ and $m_b \leq n - 3$. Since G is maximally dense, we have $m_r + m_b + m_g = 4n - 10$. It follows that $m_r + m_g \geq 3n - 7 > m_r + m_b$, which is however impossible because we are assuming $m_b \geq m_g$. Lastly, consider the case that $|f_{ext}| = 4$ and all fence faces are 3-cycles (which implies that there are exactly four fence faces because $|f_{ext}| = 4$). In every RAC drawing of G, each fence face is drawn as a triangle. Hence, for at least one of these triangles the angle opposite to the edge that belongs to f_{ext} must be larger than or equal to $\frac{\pi}{2}$. This observation, together with Lemma 2, implies that at least one of the fence faces consists of all red edges in any red-blue-green coloring. We therefore have the following: $m_r \geq (f_{rb} - 2) + \frac{|f_{ext}|}{2} + \frac{3}{2} = f_{rb} + \frac{3}{2}$. Since m_r is an integer, we have $m_r \geq f_{rb} + 2$. By $m_r + m_b \leq n + f_{rb} - 2$ we obtain $m_b \leq n - 4$, and by $m_r + m_b + m_g = 4n - 10$ we obtain $m_r + m_g \geq 3n - 6$. However, G_{rg} is a planar graph and it has the same external face as G_{rb}, that has size 4; so, G_{rg} cannot be a maximal planar graph, a contradiction. It follows that f_{ext} must be a 3-cycle. □

Lemma 4 *Graph G_{rb} is biconnected.*

Proof sketch. By Lemmas 2 and 3 the external face of G_{rb} is a 3-cycle consisting of red edges. With a similar reasoning as in the proof of Lemma 3, we obtain $m_r + m_g \geq 3n - 7$. Since $m_b \geq m_g$ we also have $m_r + m_b \geq 3n - 7$. Since G_{rb} has at least $3n - 7$ edges, it is biconnected. □

By using Lemmas 3 and 4 and a similar reasoning as in their proofs, the following lemma can be proved.

Lemma 5 *Graph G_{rb} has three fence faces. Also, each fence face of G_{rb} is a 3-cycle.*

We are now in the position of proving the following result which, together with Lemma 2, will be extensively used in the proof of Theorem 1.

Lemma 6 *G_{rb} and G_{rg} are both maximal planar graphs.*

Proof sketch. By Lemmas 3 and 5, f_{ext} is a 3-cycle consisting of red edges and the three fence faces are all 3-cycles. By simple geometric arguments it follows that in any red-blue-green coloring of a RAC drawing of G, at least two of the triangles representing these fence faces consist of red edges. We therefore have: $m_r \geq (f_{rb} - 3) + \frac{|f_{ext}|}{2} + \frac{3}{2} + \frac{3}{2}$, which implies $m_r \geq f_{rb} + 2$. By $m_r + m_b \leq n + f_{rb} - 2$, we obtain $m_b \leq n - 4$. By $m_r + m_b + m_g = 4n - 10$ we have $m_r + m_g \geq 3n - 6$.

Since G_{rg} is a planar graph, it has exactly $3n-6$ edges and so does G_{rb} because $m_b \geq m_g$. It follows that G_{rb} and G_{rg} are are both maximal planar graphs. □

3 Proof of Theorem 1

The following lemma directly implies the first part of Theorem 1.

Lemma 7 *Every RAC drawing of a maximally dense RAC graph is also a 1-planar drawing.*

Proof. Let G be a maximally dense RAC graph, let D be a RAC drawing of G and consider any red-blue-green coloring of the edges of D. Let e be a blue edge of D. By Lemma 6, every blue edge $e = (u, v)$ of G_{rb} is shared by two internal triangular faces, that we denote as f and f'. Let u, v, w be the vertices of f and u, v, w' be the vertices of f'. Since by Lemma 2 every face of G_{rb} has two red edges, we have that edges (u, w) and (w, v) are not crossed by any other edge; similarly, edges (u, w') and (w', v) of f' are both red. Since every blue edge is crossed by some green edges, we have that there can be only one green edge crossing e, namely edge (w, w'). It follows that the RAC drawing D is also a 1-planar drawing. □

To show the second part of Theorem 1, we describe an infinite family of 1-planar graphs that have the same edge density as the maximally dense RAC graphs but are not maximally dense RAC graphs. Consider first the graph G_0 of of Figure 1 (a). Clearly it is 1-planar; also, it has $n = 8$ vertices and $4n-10 = 22$ edges.

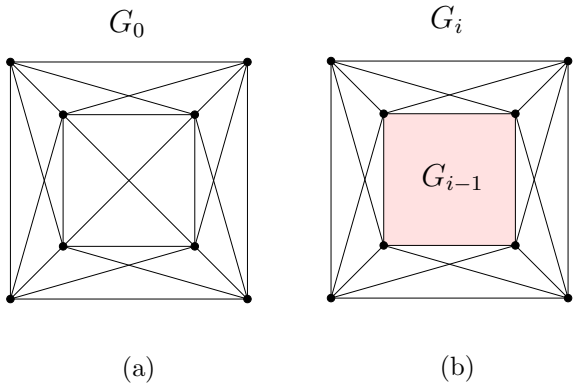


Figure 1: (a) Graph G_0; (b) Constructing graph G_i from G_{i-1}.

Lemma 8 *Graph G_0 is not a RAC graph.*

Proof. Observe thatG_0 has the following properties: (1) Every vertex of G_0 has degree at least five and at most six; (2) For every 3-cycle with vertices u, v, w, there exists a fourth vertex z such that the subgraph induced by u, v, w, z is the complete graph K_4; (3) There is a 4-cycle through the remaining four vertices, i.e. the vertices that do not form this K_4.

Suppose, for a contradiction, that G_0 had a RAC drawing D_0. By Lemma 2, the external face of D_0 is a triangle; let u, v, w be the vertices of this external face. Let z be the vertex such that the sub-drawing of D induced by vertices u, v, w, z is a planar representation of K_4. Let f_0, f_1, and f_2 be the three internal faces of this sub-drawing. Let v_0, v_1, v_2, v_3 be the remaining four vertices. They can be either all inside the same face, or they can be in two faces, or they can be in three faces. The three cases are illustrated in Figure 2.

Assume that v_0, v_1, v_2, v_3 are all in a same face, say f_0. Refer to Figure 2 (a). By Lemma 5, D_0 has three fence faces and these faces are triangles. As discussed in the proof of Lemma 6, in any red-blue-green coloring of D the edges of at least two of these three triangles are red. Since f_1 and f_2 are both fence faces, either (w, z) is a red edge or (u, z) is a red edge. Assume, w.l.o.g. that (w, z) is red. Since vertex v has degree at least five and (w, z) is red, there must be at least two edges that connect v to one of the vertices inside f_0; both such edges must cross (u, z) (see the dotted edges in Figure 2 (a)). However, by Lemma 7, D_0 is also a 1-planar drawing and (u, z) cannot be crossed twice; a contradiction.

Assume that v_0, v_1, v_2 are in f_0 and v_3 is in f_2. Refer to Figure 2 (b). Since there is a cycle with vertices v_0, v_1, v_2, v_3 there are at least two edges incident to v_3 that cross the boundary of f_2. If both these edges cross edge (u, z), then the same argument as in the previous case applies. If one of these edges crosses (v, z), it must also cross (w, z) to reach any one of v_0, v_1, v_2 (see for example the dotted edge (v_2, v_3) in Figure 2 (b)). But this would violate Lemma 7, a contradiction.

Finally, assume that v_0, v_1 are in f_0, v_2 is in f_1 and v_3 is in f_2, as depicted in Figure 2 (c). Since there is a 4-cycle with vertices v_0, v_1, v_2, v_3, there is an edge of this cycle crossing (u, z), one crossing (v, z), and one crossing (w, z). Again by Lemma 7, neither (u, z), nor (v, z), nor (w, z) can be crossed by any other edge. In order to guarantee that every vertex of G_0 has degree at least five, we must have that v_1 and v_2 are adjacent to all vertices of f_0, v_2 is adjacent to all vertices of f_1, and v_3 is adjacent to all vertices of f_3 (see the dotted edge v_2, v_3) in Figure 2 (c)). This implies that z has degree seven, which is however impossible becuase every vertex of G_0 has degree at most six.

The statement of the lemma follows. □

Lemma 8 can be generalized to prove the second part of Theorem 1.

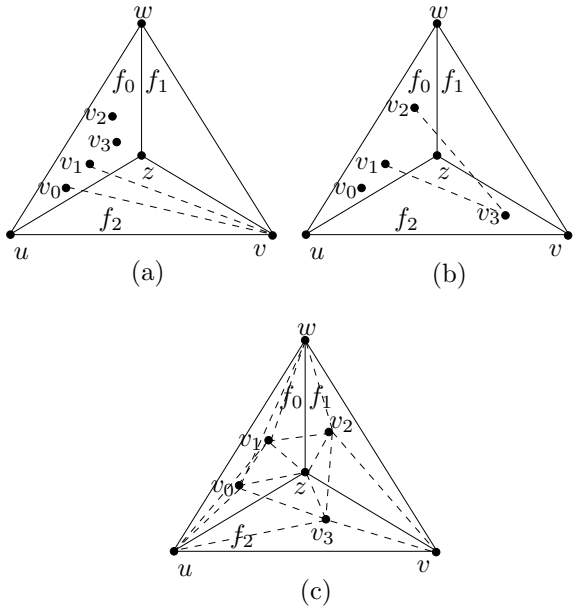


Figure 2: The three cases in the proof of Lemma 8. (a) v_0, v_1, v_2, b_3 are all in f_0, (w, z) is a red edge, and two dotted edges cross (u, z); (b) v_3 is in f_2 and edge (v_2, v_3) violates the 1-planarity condition; (c) v_3 is in f_2, v_2 in f_1 and z has degree seven.

Lemma 9 *For every integer i such that $i \geq 0$, there exists a 1-planar graph with $n = 8 + 4i$ vertices and $4n - 10$ edges that is not a RAC graph.*

Proof. Let $\mathcal{G}$ be a family of graphs defined as follows. G_0 is a graph of $\mathcal{G}$. Graph G_i of $\mathcal{G}$ is obtained from G_{i-1} by adding four vertices to the external face of G_{i-1} and 16 edges as described in Figure 1 (b). Observe that every graph in $\mathcal{G}$ is 1-planar and it has $n = 8 + 4i$ vertices and $4n - 10$ edges. Suppose that G_i had a RAC drawing D_i. Since any sub-drawing of a RAC drawing is RAC drawing too, the sub-drawing of D_i representing graph G_0 should also be a RAC drawing of G_0, contradicting Lemma 8. It follows that no graph of $\mathcal{G}$ is a RAC graph, which proves the lemma. □

Lemmas 7 and 9 prove Theorem 1.

4 Open Problems

It would be interesting to establish whether recognizing maximally dense RAC graphs is computationally as difficult as recognizing RAC graphs in the general case. Additional studies of the combinatorial properties of maximally dense RAC graphs can help in answering this question.

References

[1] E. Ackerman, R. Fulek, and C. D. Tóth. On the size of graphs that admit polyline drawings with few bends and crossing angles. In *Proc. of GD 2010*, volume 6502 of *LNCS*, pages 1–12. Springer, 2010.

[2] E. N. Argyriou, M. A. Bekos, and A. Symvonis. The straight-line rac drawing problem is np-hard. In *WALCOM*, 2011. to appear.

[3] K. Arikushi, R. Fulek, B. Keszegh, F. Moric, and C. D. Tóth. Graphs that admit right angle crossing drawings. In D. M. Thilikos, editor, *WG*, volume 6410 of *Lecture Notes in Computer Science*, pages 135–146, 2010.

[4] O. V. Borodin, A. V. Kostochka, A. Raspaud, and E. Sopena. Acyclic colouring of 1-planar graphs. *Discrete Applied Mathematics*, 114(1-3):29–41, 2001.

[5] W. Didimo, P. Eades, and G. Liotta. Drawing graphs with right angle crossings. In F. K. H. A. Dehne, M. L. Gavrilova, J.-R. Sack, and C. D. Tóth, editors, *WADS*, volume 5664 of *Lecture Notes in Computer Science*, pages 206–217. Springer, 2009.

[6] W. Didimo, P. Eades, and G. Liotta. A characterization of complete bipartite rac graphs. *Inf. Process. Lett.*, 110(16):687–691, 2010.

[7] I. Fabrici and T. Madaras. The structure of 1-planar graphs. *Discrete Mathematics*, 307(7-8):854–865, 2007.

[8] W. Huang. Using eye tracking to investigate graph layout effects. In *APVIS*, pages 97–100, 2007.

[9] W. Huang, S.-H. Hong, and P. Eades. Effects of crossing angles. In *PacificVis*, pages 41–46, 2008.

[10] M. Jünger and P. Mutzel, editors. *Graph Drawing Software*. Springer, 2003.

[11] M. Kaufmann and D. Wagner, editors. *(Eds.) Drawing Graphs*. Springer Verlag, 2001.

[12] V. P. Korzhik and B. Mohar. Minimal obstructions for 1-immersions and hardness of 1-planarity testing. In *Graph Drawing*, pages 302–312, 2008.

[13] J. Pach and G. Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997.

[14] H. C. Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with Computers*, 13(2):147–162, 2000.

[15] H. C. Purchase, D. A. Carrington, and J.-A. Allder. Empirical evaluation of aesthetics-based graph layout. *Empirical Software Engineering*, 7(3):233–255, 2002.

[16] Y. Suzuki. Optimal 1-planar graphs which triangulate other surfaces. *Discrete Mathematics*, 310(1):6–11, 2010.

[17] M. van Kreveld. The quality ratio of RAC drawings and planar drawings of planar graphs. In *Proc. of GD 2010*, volume 6502 of *LNCS*, pages 371–376. Springer, 2010.

[18] C. Ware, H. C. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002.

Searching for Radio Beacons with Mobile Agents that Perceive Discrete Signal Intensities

Henning Hasemann*† Tom Kamphans*‡ Alexander Kröller*

Abstract

In this work, we present a number of algorithms for an agent to find a radio source in an unknown two-dimensional environment. We consider a model in which a mobile agent can perceive the intensity of a radio beacon only in discrete intervals instead of a uniform signal distribution.

Our solution for the general case also allows multiple local intensity maxima. Moreover, we present algorithms that behave efficiently if the width of intensity layers is bounded. For the case without geometric obstacles we also provide a lower bound for online algorithms and show that our solution operates within a constant factor of this bound.

1 Introduction

The problem of finding a radio source with mobile agents among geometric obstacles in an unknown environment has been studied intensively. The nature of the solutions vary substantially with the sensor model that is chosen for the implementing agent. A number of approaches focuses on a model in which the agent knows its exact position relative to the source at all times, such as the well-known *bug* family of algorithms [8]. Some of these approaches also grant the agent the ability to perceive parts of its environment directly [6, 3, 7]. Others require knowledge of only the direction of or the distance to the radio source [9, 2].

All of these approaches have in common that they use rather abstract models of radio-signal distributions or presume an information source other than a radio signal. However, real radio signals often do not provide direct angular or distance-related information, but still give enough information to find a direct path to the signal source [3, 11]. The idea is that even non-isotropic intensity landscapes may provide a path to the source, which can be found by following the direction of steepest signal intensity ascent.

Some real-world signal-intensity measurement systems, such as link-quality indicators (LQI, [5]), which are commonly used in wireless sensor networks, provide only discrete signal intensities. In such a setup, a signal intensity ascent may not be measurable without extensive motion. Thus, the approaches mentioned above cannot be applied. Also, there may be multiple (local) maxima of signal intensity at locations other than the radio source, due to reflection effects [10]. By combining covering techniques with search strategies, we are able to present algorithms that deal with some of these constraints and still guarantee to find the signal source.

In Section 2, we give some basic definitions that provide the groundwork for the following sections. In Section 3, we provide an approach that imposes only few constraints on the environments and still guarantees that an agent will eventually reach the signal source. Section 4 provides a lower bound for online algorithms in so-called *nested environments* without geometric obstacles and introduces an algorithm that operates within a constant factor of this bound. In Section 5, we present an algorithm that provides a maximum–path-length guarantee for the case with geometric obstacles. For a more comprehensive description, see Hasemann [4].

2 Preliminaries

An *intensity environment* (or *environment* for short) is given by a triple $E = (\mathcal{C}, p_{\text{target}}, \iota)$: the finite *configuration space* $\mathcal{C} = \mathcal{C}_{\text{free}} \dot{\cup} \mathcal{C}_{\text{blocked}} \subseteq \mathbb{R}^2$, the position of the signal source $p_{\text{target}} \in \mathcal{C}_{\text{free}}$ and an *intensity function* $\iota : \mathcal{C} \to \{1, \ldots, n_\iota\}$ with $\iota(p_{\text{target}}) = n_\iota$ where, $n_\iota \in \mathbb{N}$ is the number of different intensities that can occur in the environment. $\mathcal{C}_{\text{blocked}}$ consists of n_O obstacles $O_1, \ldots, O_{n_O}$ bounded by Jordan curves.

The set of *intensity layers*, $\mathcal{L}$, is a disjoint dissection of $\mathcal{C}$ into regions of constant value of ι. We require that every layer, $L_i \in \mathcal{L}$, is bounded by a set of Jordan curves and $\forall j \neq i : \partial L_i \cap \partial L_j \neq \emptyset \Rightarrow j \in \{i+1, i-1\}$. Regions that have no neighbor with higher ι-value are called *hills*. We call E a *nested environment* if every layer is connected, has one hole (except the layer containing p_{target}), and every layer with higher ι-value is fully contained inside this hole. Further, there is a *layer-width bound* $d \in \mathbb{R}$, such that for any $L \in \mathcal{L}$ with a hole $\forall p_l \in L : \exists p_h \in \mathcal{C} : (\iota(p_h) > \iota(p_l)) \wedge (||p_l - p_h|| \leq d)$ holds and the layer that contains p_{target} is contained in a circle with radius d around p_{target}.

For simplicity, we assume a point-shaped agent with

*Algorithms Group, Braunschweig Institute of Technology, Germany. http://www.ibr.cs.tu-bs.de/alg

†Supp. by 7th Framework Progr. contr. 258885 (SPITFIRE)

‡Supp. by 7th Framework Progr. contr. 215270 (FRONTS)

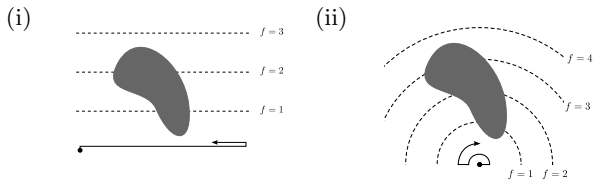


Figure 1: (i) Linear (lawn-mower–like) and (ii) circular covering patterns. Where the covering-motion curve (dotted lines) 'touches' an obstacle or an area of higher intensity, the configuration space is split into cells that are covered separately.

the capability to measure all signal intensities and to detect walls and the signal source in a radius r around its position. We assume that the agent has knowledge about whether or not the environment is nested and its layer width bound d (if any). However, the agent initially has no knowledge of the configuration space, the position of the signal source, the number of intensity layers, or the maximum intensity.

3 General Environments

We now present an algorithm that guarantees to find the signal source in general environments, using a simple back-and-forth covering strategy (e.g., sweep from left to right and back; see Fig. 1) as parameter. Note that such a strategy can cover only certain simply-connected areas, but in general is not sufficient to cover an unknown environment with obstacles and differing signal intensities. The covering pattern has to fulfill some criteria in order to make it usable for our algorithm and avoid certain corner cases. See the thesis on this subject [4] and the covering algorithm for environments without intensities [1] for details.

Algorithm 1: Intensity-Aided Coverage Planning (IACoP)

The agent starts in the outer layer (i.e., $\iota($ current position $) = 1$).
Cover the current region using the given pattern, until an obstacle or an area of higher intensity is found. In this case, add the newly-found areas to a list. From this list, choose one of the areas with highest intensity and proceed with this area. If an area is completely covered, remove it from the list. Stop when p_{target} is found.

It is easy to see that the IACoP strategy finds the signal source, as it eventually covers $\mathcal{C}_{\text{free}}$ completely. Although this algorithm has the upside of being very general in terms of constraints imposed on the environment, it can be forced to cover an area that is arbitrarily large in comparison to the agent's distance to the source. In the following section, we will see that nested environments can be searched more efficiently.

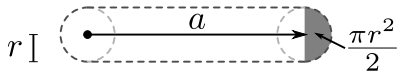


Figure 2: $A_{\text{tube}}(a)$

4 Nested Environments without Obstacles

In this section, we consider nested environments with a layer width bound d. We provide a lower bound on the agent's path length, and an algorithm for the case $\mathcal{C}_{\text{blocked}} = \emptyset$; the case with obstacles is discussed in the following section.

4.1 Lower Bound

Lemma 1 *Let $A \subset \mathbb{R}^2$ be an area with $|A| < \infty$. A path of length $l_{\text{covmin}}(|A|) = \min\{(|A| - \pi r^2)/2r, 0\}$ is always necessary and sometimes sufficient for an agent with coverage radius r to cover A completely.*

Proof. Assume that there is an area A with $\pi r^2 \leq |A| < \infty$ that is coverable by a path of length $l < l_{\text{covmin}}(|A|) = \frac{|A|-\pi r^2}{2r}$. Initially (i.e., prior to any actual motion), the agent covers $|A_{\text{init}}| \leq \pi r^2$. Then, during each motion step of length ε, the agent covers an additional area A_ε with $|A_\varepsilon| = \frac{|A|-|A_{\text{init}}|}{l}\varepsilon$ on average. We observe that $|A_\varepsilon| > 2r\varepsilon\frac{|A|-|A_{\text{init}}|}{|A|-\pi r^2} \geq 2r\varepsilon$ holds. However, for a step of length ε, the agent cannot do better than moving on a straight line into uncovered space. Such a step cannot cover more than $2r\varepsilon$ which is a contradiction to $|A_\varepsilon| > 2r\varepsilon$. Thus, an agent cannot cover any area A with a path shorter than $l_{\text{covmin}}(|A|)$.

The area $A_{\text{tube}}(a)$, see Fig. 2, with $|A_{\text{tube}}(a)| = 2ra + \pi r^2$ can clearly be covered by an agent moving on a straight line of length $a = (2ra + \pi r^2 - \pi r^2)/2r = l_{\text{covmin}}(|A_{\text{tube}}(a)|)$. Thus, a path of length $l_{\text{covmin}}(|A|)$ is sometimes sufficient for covering an area A. □

Theorem 2 *For any online algorithm* ONL *that finds the signal source in a nested environment without obstacles, there exists an environment E with layer width bound $d \geq 2r$ and $n_\iota \geq 3$ intensity layers, such that an agent with a coverage radius of r executing ONL in E travels a path of length* $D_{\text{ONL}} \geq (n_\iota - \frac{1}{2})d + \frac{\pi d^2}{2r} + (\frac{1}{2} - \frac{3\pi}{4})r$.

Proof. We construct E, such that an agent must walk a path of length $(n_\iota - 1)d$ in order to reach the hill. Let p_{n_ι} be the first point of the hill that is covered by the agent. We place the signal source at the point in the disk of radius d around p_{n_ι} that is last to be covered by the agent. Note that the agent must already have covered (slightly less than)

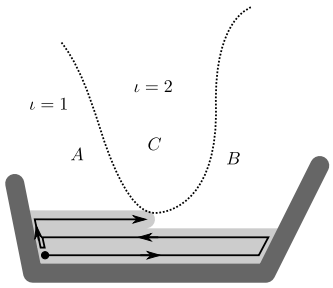


Figure 3: A mobile agent covering the environment. When detecting an area of different intensity, the agent creates three new cells A, B and C. The one with the highest intensity (C) will be covered first.

$|A_{\text{tube}}(d-r)| - \frac{\pi}{2}r^2$ of that disk, so the length of the path an agent travels after it has detected p_{n_ι} in E is at least $l_{\text{covmin}}(|A_{\text{tube}}(d-r)| - \frac{\pi}{2}r^2)$.

Altogether, the agent has to travel a distance of at least $(n_\iota - \frac{1}{2})d + \frac{\pi d^2}{2r} + (\frac{1}{2} - \frac{3\pi}{4})r$ in order to reach the signal source. □

We conjecture that it is possible to construct a better lower bound by exploiting the fact that the mobile agent cannot know in which direction an area of higher intensity is located. However, it is difficult to predict a minimal path length, because an optimal path strongly depends on the shape of the areas that the agent has already covered.

4.2 Circular Motion Planning

Algorithm 2: Circular Motion Planning (CMoP)

```
c_1 := current position;
i := 1;
walk an arbitrary straight line of length d − r;
repeat
    walk on a circle around c_i;
    on intensity increase detected at some point p
        i := i + 1;
        c_i := p;
        go to nearest point q with ‖c_i − q‖ = d − r;
until circle complete;
move on concentric circles towards c_i;
```

Theorem 3 *An agent executing the CMoP algorithm in a nested environment will reach the radio source with a total maximum travel distance of* $D_{\text{CMoP}} \leq 8.192\, D_{\text{ONL}}$.

Proof. When the agent starts executing the algorithm, it travels a straight line of length $d - r$, followed by at most a complete circle of length $2\pi(d-r)$ around c_1.

For the following circles around a point c_k, $k \in \{2, \ldots, n_\iota - 1\}$ a point p of intensity $\iota(p) > \iota(c_k)$ can either be found during circling or not. If it is found, a new circle is started around $p = c_{k+1}$ and the way costs for the circle around c_k are $d - 2r$ for getting to the closest point on the circle and at most $2\pi(d-r)$ for the actual circular motion.

If the agent has not found a point with higher signal intensity than $\iota(c_k)$ so far, such a point must be inside the d-disk around c_k. As the environment is nested, the same holds true for all other points with higher intensity than $\iota(c_k)$, especially the radio source.

In the last step, we walk on $\lceil \frac{d}{2r} \rceil$ concentric circles towards c_k and need a total of d for the relocation between two successive cirlces. From the last circle, we move a distance of r towards the source:

$$\begin{aligned} D_{\text{CMoP}} &\leq (2\pi+1)(d-r) + ((2\pi+1)d - (2\pi+2)r)(n_\iota - 2) \\ &\quad + \textstyle\sum_{i=1}^{\lceil \frac{d}{2r} \rceil} 2\pi r(2i-1) + d + r \\ &\leq (2\pi+1)\,(n_\iota - 2)\,(d-r) + \tfrac{\pi d^2}{2r} + 4\pi d + 2 \\ &< (2\pi+1) \cdot \tfrac{n_\iota}{n_\iota - \frac{1}{4} - \frac{3}{8\pi}} \cdot D_{\text{ONL}} \\ &\leq 8.192\, D_{\text{ONL}} \end{aligned}$$ □

We observe that for high values of n_ι, this bound approaches $2\pi + 1$ (≈ 7.2832).

5 Nested Environments with Obstacles

For nested environments with obstacles, we use the Circular Coverage Planning Algorithm (CCoP), see Alg. 3. Figure 4 shows an example of this algorithm.

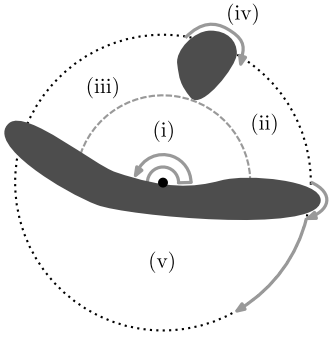


Figure 4: Example run of the CCoP algorithm. The agent starts covering area (i) with a circular pattern until it detects an obstacle (dashed grey line). Next, areas (ii) and (iii) are covered separately. Then the agent follows the nearest obstacle wall (iv) and marks the obstacle as covered. The agent follows another obstacle wall and covers the newly found part of the d-disk (v) starting with large circles.

Theorem 4 *An agent executing the CCoP algorithm will reach the radio source with a total maximum*

Algorithm 3: Circular Coverage Planning Algorithm (CCoP)

Let p be the agent's starting position.

- Walk a concentric circle pattern around p.
- Whenever a wall piece is detected, store its position in a list. Also, store the positions of the sub-areas to both sides of the wall piece.
- When the current circle cannot be continued (either because it is complete or because of an obstacle), recursively cover all known but not covered cells by extending the circular pattern to those areas.
- When the current circle radius has reached d, no point of higher intensity can be found, and all known cells are covered, follow the nearest wall piece until an area of higher intensity or an uncovered part of the d-circle around p is found. In the latter case, cover this part, then continue following the wall so that it will eventually be circled completely.
- Whenever an obstacle has been circled completely, mark the area of the obstacle as already covered.
- Whenever a point p' with intensity higher than $\iota(p)$ is detected, move to p and start the algorithm from the beginning with $p := p'$.
- When the signal source is detected, move to the source, and stop.

travel distance of
$D_{\mathrm{CCoP}} \leq (\frac{\pi d^2}{2r}+(2\pi+1)d+(2\pi+2)r)n_\iota+2\sum_{i=1}^{n_O}|\partial O_i|$.

Proof. The agent starts with a covering loop around its starting position. The actual covering happens in $\lceil \frac{d}{2r} \rceil$ circles with radii $r, 3r, 5r, \ldots \left(2\lceil \frac{d}{2r} \rceil - 1\right) r$. Then the circular part of the covering loops is bounded by $\sum_{i=1}^{\lceil \frac{d}{2r} \rceil} ((2i-1)2\pi r)$.

For relocation between covering loops, the agent walks a distance of at most d when there are no obstacles. In addition, the agent will walk n_ι path segments, each of length no more than r, for getting from one spiral to the next or to the source.

When there are obstacles that intersect the covering loop, the relocations happen on the obstacle boundaries, and thus are bounded by $\sum_{i=1}^{n_O}|\partial O_i|$. The search for an unexplored part of the covering circle is also a path along obstacle borders, so its length is bounded by $\sum_{i=1}^{n_O}|\partial O_i|$, too.

In total we get

$$\begin{aligned} D_{\mathrm{CCoP}} &\leq \left[\sum_{i=1}^{\lceil \frac{d}{2r} \rceil} ((2i-1)2\pi r) + 2\left\lceil \tfrac{d}{2r} \right\rceil r\right] n_\iota \\ &\quad + 2\sum_{i=1}^{n_O}|\partial O_i| \\ &\leq \left[(2\pi+1)d + \tfrac{\pi}{2}\tfrac{d^2}{r} + (2\pi+2)r\right] n_\iota \\ &\quad + 2\sum_{i=1}^{n_O}|\partial O_i| \end{aligned}$$

□

6 Conclusion

We presented approaches for three different cases of environments with discrete signal intensities. In the case of environments with a layer width bound and no obstacles we could show that our algorithm is within a constant factor of an optimal online algorithm.

Acknowledgements

We thank Sándor P. Fekete for helpful discussions and proofreading.

References

[1] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull. Morse decompositions for coverage tasks. *Internat. J. Robot. Res.*, 21(4):331–344, 2002.

[2] D. Angluin, J. Westbrook, and W. Zhu. Robot navigation with distance queries. *SIAM J. Comput.*, 30:110–144, 2000.

[3] M. Burger, Y. Landa, N. Tanushev, and R. Tsai. Discovering point sources in unknown environments. In *Proc. Workshop Algorithmic Found. Robot.*, 2008.

[4] H. Hasemann. Searching for radio beacons among geometric obstacles with mobile agents. Master's thesis, Braunschweig Institute of Technology, 2010. http://www.ibr.cs.tu-bs.de/users/hasemann/.

[5] IEEE. Standard 802.15.4 for local and metropolitan area networks specific requirements part 15.4, 2006.

[6] I. Kamon and E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Trans. Robot. Autom.*, 13:814–822, 1997.

[7] V. J. Lumelsky and T. Skewis. A paradigm for incorporating vision in the robot navigation function. In *Proc. IEEE Internat. Conf. Robot. Automat.*, pages 734–739, 1988.

[8] V. J. Lumelsky and A. A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[9] V. J. Lumelsky and S. Tiwari. An algorithm for maze searching with azimuth input. In *Proc. IEEE Internat. Conf. Robot. Automat.*, pages 111–116, 1994.

[10] L. Tang, K.-C. Wang, Y. Huang, and F. Gu. Channel characterization and link quality assessment of ieee 802.15.4-compliant radio for factory environments. *IEEE Trans. Industrial Informatics*, 3(2):99–110, 2007.

[11] K. Taylor and S. M. LaValle. I-bug: An intensity-based bug algorithm. In *Proc. IEEE Internat. Conf. Robot. Automat.*, pages 3981–3986, 2009.

Area-Preserving $\mathcal{C}$-Oriented Schematization*

Kevin Buchin† Wouter Meulemans† Bettina Speckmann†

Abstract

We define an *edge-move* operation for polygons and prove that every simple non-convex polygon P has a non-conflicting pair of complementary edge-moves that reduces the number of edges of P while preserving its area. We use this result to generate area-preserving $\mathcal{C}$-oriented schematizations of polygons.

1 Introduction

A schematic map displays a set of nodes and their connections—for example, highway, train, or metro networks—in a highly simplified form to communicate the connectivity information as effectively as possible. Connections are usually drawn as polygonal paths using few links and few orientations. The set of permissible orientations often contains only the two axis-parallel or the four main orientations. The most general setting is $\mathcal{C}$-oriented schematization where every link has to use one of a set $\mathcal{C}$ of specified orientations.

A substantial part of previous efforts concentrates on the schematization of networks, but it is also often desirable to schematize the boundaries of regions or even complete subdivisions. Whenever exact boundaries are not needed it is preferable to replace them by schematic ones, to reduce visual clutter and to indicate that the map is not a topographic map.

The schematized output should visually resemble the input. But it is not clear how to quantify what the most recognizable schematization of a given input is. Optimization based on standard distance metrics can produce undesirable output for certain input polygons [8]. Hence we focus on area-preserving schematization, that is, the area of the input polygon and its schematization are equivalent. While we do not prove any guarantees on the resulting shapes, experimental results are quite promising and visually pleasing.

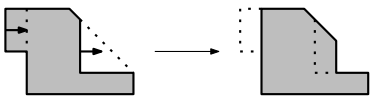


Figure 1: Complementary edge-moves.

*W. Meulemans and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.022.707.

†Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands. k.a.buchin@tue.nl, w.meulemans@tue.nl, and speckman@win.tue.nl.

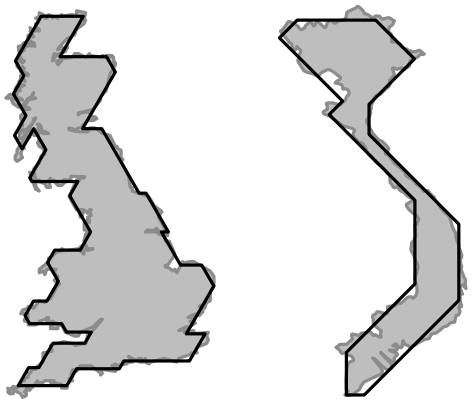


Figure 2: Hexagonal Great Britain using 50 edges; octagonal Vietnam using 15 edges.

Results. We introduce an *edge-move* operation that moves an edge of a polygon inward or outward without changing its orientation. This does change the length of its adjacent edges and potentially itself. A contraction is an edge-move that reduces at least one edge to length zero, hereby reducing the number of edges in the polygon. Since we desire area preservation, we apply this operation in non-conflicting complementary pairs (Figure 1). Theorem 1, our main result, is an immediate consequence of Lemma 6:

Theorem 1 *Every simple non-convex polygon has a non-conflicting pair of complementary edge-moves, one of which is a contraction.*

As edge-moves do not introduce new orientations, we can also use edge-moves to create area-preserving $\mathcal{C}$-oriented schematizations of polygons. Note that a simple polygon can be converted into a simple $\mathcal{C}$-oriented polygon of equal area based on the method presented by Meulemans *et al.* [8].

Corollary 2 *Given a simple $\mathcal{C}$-oriented polygon P and an integer k with $2|\mathcal{C}| \leq k$, an area-preserving $\mathcal{C}$-oriented schematization of P with at most k edges can be generated using only non-conflicting pairs of complementary edge-moves.*

The results in Figure 2 were obtained by repeatedly executing a non-conflicting pair of complementary edge-moves such that the contraction minimizes the area change and the compensating edge-move is as close by as possible. This method can also be used for simple subdivisions restricted to edge-moves that do

not change the topology. Our algorithm then ensures that each face preserves its area and that adjacencies are correct. In this case we cannot give any guarantees on the reduction in the number of edges. Preliminary experiments suggest that the reduction is significant.

Related work. There is an ample body of work on map schematization and metro map construction, see the surveys by Swan *et al.* [10] and Wolff [11] for an overview. Of particular interest is the work by Merrick and Gudmundsson [7] who describe an algorithm to generate $\mathcal{C}$-oriented metro map layouts. Methods for map schematization can be used to schematize subdivisions, but they usually do not take criteria such as shape and size preservation into account.

The generalization of urban data, specifically building generalization, is closely related to our work. In particular, algorithms for building wall squaring [6] and outline simplification [5] can be used for polygon schematization and vice versa.

Line simplification has been a prominent topic in the GIS literature for many years. Of particular relevance are the work of Delling *et al.* [3] and Neyer [9] dealing with $\mathcal{C}$-oriented schematization of routes or lines and the work of Bose *et al.* [1] on area-preserving line simplification. However, it is generally not advisable to schematize each chain in a subdivision separately. There are some approaches, developed in computational geometry, that preserve the topology of the input subdivision. For example, De Berg *et al.* [2] describe a method that simplifies a polygonal subdivision without introducing intersections or passing over special input points. Unfortunately many subdivision simplification problems that minimize the number of edges in the output are NP-complete [4].

2 Edge-moves and configurations

Definitions and notation. We are given a simple polygon P with vertices $v_1, \ldots, v_n$. We treat the vertices modulo n, e.g. $v_{n+1} = v_1$. The edges are denoted by $e_1, \ldots, e_n$, again treated circularly. The directed edge e_i starts at vertex v_i and ends at vertex v_{i+1}. A vertex is called convex if the angle inside the polygon between its two adjacent edges is at most π, and it is called reflex otherwise. We call an edge convex or reflex if both its vertices are convex or reflex respectively. The exterior angle of a vertex is defined as the angle between one edge and the extension of the other. The exterior angle is sometimes also referred to as turning angle. The angle is negative if and only if the vertex is reflex. The sum of all exterior angles of a simple polygon is always equal to 2π.

We define a chain S as a set of at least three consecutive edges of P. Its edges are denoted by $s_1, \ldots, s_m$ with $m \leq n$. Its vertices are denoted by $u_1, \ldots, u_{m+1}$ and edge s_i is directed from u_i to u_{i+1}. The edges

Figure 3: A positive and negative exterior angle.

s_1 and s_m are the *outer edges* of S, the other edges are its *inner edges*. Likewise, u_1 and u_{m+1} are outer vertices, the other vertices inner vertices. By $\alpha(S)$, we denote the sum of the exterior angles of the inner vertices of S. A *lid* is an open line segment between a point on s_1 and a point on s_m and is fully contained in the interior of P. If S has any lid, it is a *closable chain*. If the open line segment (u_1, u_{m+1}) is a lid, S is a *proper chain*. For a closable chain S and a lid l, we denote by $R_l(S)$ the region enclosed by S and l. For a proper chain, $R(S)$ denotes this region using the lid (u_1, u_{m+1}) implicitly. Due to the lid, we know that for every closable chain, any point on the boundary of P that is inside $R_l(S)$ must be part of S. For any closable chain, $\alpha(S) > 0$ holds. A (not necessarily closable) chain with exactly 3 edges is called a *configuration* G, its edges denoted by g_1, g_2, and g_3.

The outer edges of a configuration G define two tracks, infinite lines through the edges. An *edge-move* on G moves g_2 such that its orientation is preserved and its vertices are on the tracks, making the outer edges longer or shorter. An edge-move is valid if at least one of its vertices remains on its original outer edge and g_2 remains on the same side of or on the intersection point of the tracks (if any). An edge-move which causes one of the edges of G to reach length zero, is a *contraction*. Contractions are extremal edge-moves. An edge-move is *positive* if it adds area to P and *negative* if it removes area.

A configuration supports edge-moves, either positive, negative or both. Let g_2^+ denote the extremal position of g_2 after any valid positive edge-move, i.e. the position after a positive contraction. The *positive contraction region* of a positive configuration, $R^+(G)$, is the region enclosed by g_2, g_2^+, and the tracks. A *feasible positive configuration* is a configuration for which $R^+(G)$ is empty except for G. Similarly, we define the *negative contraction region* $R^-(G)$ and a feasible negative configuration. If a positive or negative configuration is feasible, then any valid positive or negative edge-move respectively is feasible. If a positive configuration is infeasible, then there is some point on δP in $R^+(G) \backslash G$. A point in $\delta P \cap R^+(G) \backslash G$ that is closest to

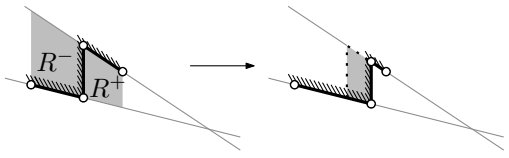


Figure 4: A valid positive edge-move.

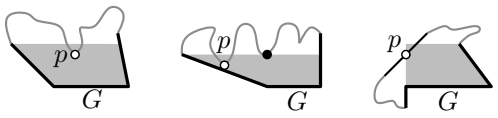


Figure 5: Configurations G and blocking points p.

the line through g_2 is called a *positive blocking point*. Analogously, G can have a *negative blocking point*. If the inner edge of an infeasible G is convex or reflex, there is always a blocking point that is a vertex of P.

Since we desire an approach that preserves the area of the polygon, we combine two *complementary* feasible configurations, one positive and one negative, executing an edge-move on both simultaneously. The one with the smaller contraction region is contracted, while the other is moved just far enough to compensate for the area change. Two configurations *conflict* when they share an edge, unless they share only outer edges and one of these has a convex and a reflex vertex. In this special case the two edge-moves both either shorten or lengthen the shared edge. We call two non-conflicting complementary feasible configurations a *proper configuration pair*.

Completeness. We now prove that any non-convex polygon has a proper configuration pair. First we discuss some properties of a closable or proper chain.

Lemma 3 *If S is a closable chain without a convex inner edge and s_2 is reflex, then S has a feasible negative configuration G with a reflex first inner vertex, $\alpha(G) > 0$ and $R^-(G) \subseteq R_l(S)$ for any lid l of S.*

Proof. As $\alpha(S) > 0$ and there are no convex inner edges, there must a configuration G' in S with a reflex first inner vertex and $\alpha(G') > 0$ (implying that the second inner vertex is convex). Let G' be the first such configuration. A configuration G'' with $\alpha(G'') > 0$ and a convex first inner vertex may occur multiple times before G', but it must always be preceded by an edge g_0'' with $\alpha(\{g_0''\} \cup G'') < 0$ for the chain $\{g_0''\} \cup G''$ as otherwise G' is not the first of its kind along S.

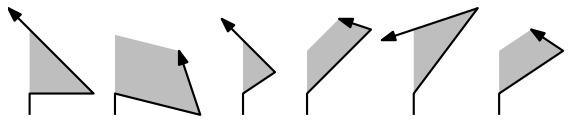


Figure 6: Configurations with $\alpha > 0$ and a reflex first inner vertex.

We prove this lemma by induction. Assume that $m = 3$. Chain S is a configuration G' with a reflex first inner vertex. Any lid l of S must be on one side of the line through s_1, whereas this track enforces a triangular contraction region on the other side. Hence, $R^-(G') \subseteq R_l(S)$ holds and G' is a feasible negative configuration. Examples 1, 3 and 5 in Figure 6 show closable chains, the other examples do not.

For induction, assume that this lemma holds for any closable chain with less than m edges and with a reflex first inner vertex. Let $G' = s_{i-1}s_is_{i+1}$ be the first configuration with $\alpha(G') > 0$ where u_i is reflex. If G' is feasible, we are done. If G' is not feasible, let p denote the corresponding blocking point. Note that p must be part of S as $R^-(G') \subseteq R_l(S)$ for any lid l. Moreover, p must come after s_{i+1} on S as otherwise G' cannot be the first. The closable chain $S' = s_{i+1}, \ldots, p$ (that is, until the edge containing p if p is not a vertex) has less edges than S and thus has a feasible negative configuration by induction. □

Lemma 4 *If a proper chain S has a convex inner edge, then S has a feasible negative configuration G with $R^-(G) \subseteq R(S)$ and $\alpha(G) > 0$. Also, g_2 is convex or starts at a reflex vertex or $g_2 \neq s_{m-1}$.*

Proof. We prove this lemma by induction on m. If $m = 3$, S is a feasible negative configuration, since S is a proper chain and s_2 is a convex edge. For induction, assume that this lemma holds for any suitable chain with less than m edges. Let s_i be a convex inner edge. Hence, $G' = s_{i-1}s_is_{i+1}$ is a negative configuration. If G' is feasible, we are done as a convex edge implies $\alpha(G') > 0$. If G' is not feasible, then let u_j denote the blocking vertex and assume without loss of generality that $i + 1 < j$. Note that u_j must be a vertex of the chain as $R^-(G') \subseteq R(S)$. Consider the proper chain $S' = s_i, \ldots, s_{j-1}$. If S' has a convex inner edge, it must have a feasible negative configuration by induction. However, if it does not, $S'' = s_{i+1}, \ldots, s_{j-1}$ is a closable chain in which s_2' is reflex. Note that the direction of S'' is reversed when $j < i-1$. By Lemma 3, S'' has a feasible negative configuration G''. By the lemma, the first inner vertex of G'' is reflex in the direction of S''. If S'' was reversed, then $s_m \notin S''$, thus s_{m-1} cannot be the inner edge of G''. □

Lemma 5 *Every simple non-convex polygon P has a feasible positive configuration G with $\alpha(G) < 0$ or all positive configurations are feasible.*

Proof. If P has a reflex edge, then let G be a configuration with a reflex inner edge. If G is feasible, we are done as $\alpha(G) < 0$. If it is not, we can define a chain $\overline{S}$ that is inverted: the interior of $\overline{S}$ is in fact the exterior of the polygon. Using Lemma 3 or Lemma 4, we can now find a feasible negative configuration G^- with $\alpha(G^-) > 0$ in $\overline{S}$. This corresponds to a feasible positive configuration G^+ in P with $\alpha(G^+) < 0$.

If P has no reflex edge, then let G be an infeasible positive configuration. If none exists, all positive configurations are feasible. Otherwise we can define an inverted chain using G. Thus, by Lemma 3, P has a feasible positive configuration G' with $\alpha(G') < 0$. □

Lemma 6 *Every simple non-convex polygon P has a proper configuration pair.*

Proof. By Lemma 5, polygon P has a feasible positive configuration $G^+ = e_{i-1}e_ie_{i+1}$. Assume without loss of generality that the second inner vertex of G^+, v_{i+1}, is reflex. Let v_j denote the first convex vertex after v_{i+1}. Configuration $G^- = e_{j-1}e_je_{j+1}$ of which v_j is the first inner vertex, is negative. We distinguish two cases.

Assume that G^- is feasible. If no edge is shared or if edge $e_{i+1} = e_{j-1}$ is shared (having a convex and a reflex vertex), we have a proper configuration pair. If $e_{i-1} = e_{j+1}$ is shared but the other outer edge is not, then $v_j, v_{j+1} = v_{i-1}, v_i$ are the only convex vertices in P and there is at least one edge in between e_i and e_{j-1}. This edge is the inner edge of a feasible positive configuration, one that does not conflict with G^-.

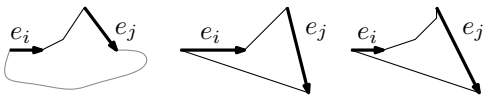


Figure 7: Three cases if G^- is feasible.

Now, assume that G^- is not feasible. The blocking point cannot be in between v_i and v_{j+1}. If G^- is blocked by a vertex v_h, then, depending on the convexity of v_{j+1} and v_{j+2}, either Lemma 3 or Lemma 4 shows that there is a (non-conflicting) feasible negative configuration. If G^- is blocked by an edge e_h, v_{j+1} must be reflex. We distinguish two cases on the closable chain $S = e_j, \ldots, e_h$.

If S does not have a convex inner edge, then we refer to Lemma 3 to find a feasible negative configuration G'. If G' does not conflict with G^+, we are done. If G' does conflict with G^+, we know that $e_h = e_{i-1}$ holds and that e_{h-1} is the inner edge of G'. Moreover, it must now hold that $\alpha(G^+) > 0$ and thus, by Lemma 5, we need to consider this case only when all positive configurations are feasible. Hence, the positive configuration $e_ie_{i+1}e_{i+2}$ is feasible and it does not conflict with G'.

Figure 8: Two cases if G^- is not feasible and S has no convex inner edge. G' may conflict with G^+.

If S has a convex inner edge e_t, then let G' denote the negative configuration $e_{t-1}e_te_{t+1}$. If G' is feasible and not conflicting with G^+, we are done. If G' is feasible but conflicting with G^+, we can argue as above: $e_ie_{i+1}e_{i+2}$ is a feasible positive configuration and it does not conflict with G'. If G' is not feasible, then it must be blocked by some vertex v_b. Without loss of generality, assume that the proper chain $S' = e_t, e_{t+1}, \ldots, e_{b-1}$ does not contain edge e_i. Depending on the convexity of vertex v_{t+2}, Lemma 3 or Lemma 4 shows that there is a feasible negative configuration G'' in S'. We now argue why G'' cannot conflict with G^+. The only way to have a conflict is when $v_b = v_i$. Since v_i is then reflex, the only way to obtain a conflict is when v_{i-1} is reflex as well and v_{i-2} is convex, such that $e_{i-3}e_{i-2}e_{i-1}$ is a negative configuration. However, e_{i-2} is the before-last edge in S' and starts at a convex vertex. Hence, Lemma 3 and Lemma 4 guarantee to find another feasible negative configuration instead. □

References

[1] P. Bose, S. Cabello, O. Cheong, J. Gudmundsson, M. van Kreveld, and B. Speckmann. Area-preserving approximations of polygonal paths. *Journal of Discrete Algorithms*, 4(4):554–566, 2006.

[2] M. de Berg, M. van Kreveld, and S. Schirra. Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and Geographic Information Science*, 25(4):243–257, 1998.

[3] D. Delling, A. Gemsa, M. Nöllenburg, and T. Pajor. Path schematization for route sketches. In *Proc. 12th Scandinavian Workshop on Algorithm Theory*, LNCS 6139, pages 285–296, 2010.

[4] L. Guibas, J. Hershberger, J. Mitchell, and J. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry and Applications*, 3:383–415, 1993.

[5] J. Haunert and A. Wolff. Optimal and topologically safe simplification of building footprints. In *Proc. 18th International Conference on Advances in GIS*, pages 192–201, 2010.

[6] H. Mayer. Scale-spaces for generalization of 3d buildings. *International Journal of Geographical Information Science*, 19:975–997, 2005.

[7] D. Merrick and J. Gudmundsson. Path simplification for metro map layout. In *Proc. 14th International Symposium on Graph Drawing*, LNCS 4372, pages 258–269, 2006.

[8] W. Meulemans, A. van Renssen, and B. Speckmann. Area-preserving subdivision schematization. In *Proc. 6th International Conference on GIScience*, LNCS 6292, pages 160–174, 2010.

[9] G. Neyer. Line simplification with restricted orientations. In *Proc. 6th International Workshop on Algorithms and Data Structures*, LNCS 1663, pages 13–24, 1999.

[10] J. Swan, S. Anand, M. Ware, and M. Jackson. Automated schematization for web service applications. In *Web and Wireless Geographical Information Systems*, LNCS 4857, pages 216–226, 2007.

[11] A. Wolff. Drawing subway maps: A survey. *Informatik-Forschung und Entwicklung*, 22(1):23–44, 2007.

New bivariate system solver and topology of algebraic curves

Yacine Bouzidi* Sylvain Lazard* Marc Pouget* Fabrice Rouillier†

Abstract

We present a new approach for solving polynomial systems of two bivariate polynomials with rational coefficients. We first use González-Vega and Necula approach [3] based on sub-resultant sequences for decomposing a system into subsystems according to the number of roots (counted with multiplicities) in vertical lines. We then show how the resulting triangular subsystems can be efficiently solved by computing lexicographic Gröbner basis and Rational Univariate Representations (RURs) of these systems. We also show how this approach can be performed using modular arithmetic, while remaining deterministic.

Finally we apply our solver to the problem of computing the topology of algebraic curves using the algorithm Isotop [2]. We show that our approach yields a substantial gain of a factor between 1 to 10 on curves of degree up to 28 compared to directly computing a Gröbner basis and RUR of the input system, and how it leads to a very competitive algorithm compared to the other state-of-the-art implementations.

1 Bivariate system solving algorithm

We present in this section our new bivariate solver and discuss in Section 2 its application to the computation of the topology of plane algebraic curves.

The input is a system $S = \{P, Q\}$ where P and Q are two bivariate polynomials in $\mathbb{Q}[x, y]$. We solve the system, that is the output is a set of boxes isolating the solutions of S, i.e., pairwise-disjoint axis-parallel boxes in the xy-plane, each containing exactly one root.

Our algorithm proceeds in two steps. First we decompose the system S into a set of triangular systems according to the sub-resultant sequence of P and Q (Section 1.1). This decomposition is essentially identical to that of González-Vega and Necula [3]; for completeness, we describe this decomposition in Section 1.2 together with our extension for the treatment of solutions for which the leading coefficient of P or Q (seen as a polynomial in y) vanishes. In the second step, we compute a Gröbner basis and a Rational Univariate Representation (RUR) [4] of every system of the decomposition (Sections 1.3). We show how the specific form of the input triangular systems permits to trivially compute a Gröbner basis, and yields improvements to the algorithm of [4] for computing a RUR. The RUR is then used in a standard way to solve these systems.

We show in Section 1.4 how these two steps can be performed with modular arithmetic in a deterministic way, yielding a very efficient algorithm.

It should be stressed that we do not compute the multiplicity of the roots in the input system. However, for every root (α, β) of a system $\{F, \frac{\partial F}{\partial y}\}$, we obtain the multiplicity of β as a root of $F(\alpha, y)$ (Section 2). These multiplicities are needed in our algorithm for computing the topology of plane curves, and this is a key feature of our solving algorithm.

1.1 Preliminaries : sub-resultant sequences

Recall that P and Q are two polynomials in $\mathbb{Q}[x, y]$. Considering P and Q as polynomials in y, let $P = \sum_{i=0}^{p} a_i y^i$, and $Q = \sum_{i=0}^{q} b_i y^i$ with a_i, b_i in $\mathbb{Q}[x]$. Assume without loss of generality that $p \leqslant q$.

The Sylvester matrix of P and Q is the $(p+q) \times (p+q)$ matrix where the k-th row consists, for $1 \leqslant k \leqslant q$, of $k-1$ zeros, followed by the coefficients of P, $a_p, \ldots, a_0$, and completed by $q-k$ zeros, and, for $q+1 \leqslant k \leqslant p+q$, of $k-q-1$ zeros, followed by the coefficients of Q, $b_q, \ldots, b_0$, and completed by $p+q-k$ zeros. One denotes by $Sylv_i$, $i \leqslant q$, the $(p+q-2i) \times (p+q-i)$ matrix obtained from the Sylvester matrix of P and Q by deleting the i last rows of the coefficients of P, the i last rows of the coefficients of Q, and the i last columns.

Definition 1 ([1]) *The i-th polynomial* sub-resultant *of P and Q, denoted by $Sres_i(P, Q)$, is the polynomial*

$$\det(M_{p+q-2i})y^i + \det(M_{p+q-2i+1})y^{i-1} + \cdots + \det(M_{p+q-i})$$

where M_j is the square sub-matrix of $Sylv_i$ of size $p+q-2i$ and consisting of the $p+q-2i-1$ first columns and the j-th column of $Sylv_i$, $j \in \{p+q-2i, \ldots, p+q-i\}$. The sub-resultant sequence *of P and Q is the sequence $Sres_i(P, Q)$ for i from 0 to q.*

$Sres_i(P, Q)$ is a polynomial of degree at most i in y, and the coefficient of the monomial of degree i, denoted by $sres_i(P, Q)$, is called the i-th *principal sub-resultant coefficient*. Note that $Sres_0(P, Q) =$

*INRIA Nancy Grand Est, LORIA laboratory, Nancy, France. Firstname.Name@loria.fr

†INRIA Paris-Rocquencourt and LIP6 (Université Paris 6, CNRS), Paris, France. Firstname.Name@lip6.fr

$sres_0(P,Q)$ is the *resultant* of P and Q. We thus have

$$Sres_i(P,Q) = sres_i(P,Q)y^i + R_i(x,y) \qquad (1)$$

where the degree in y of R_i is at most $i-1$, and its degree in x is at most the product of the total degrees of P and Q (the degree in x is generically maximal for $i=0$ which is that of the resultant).

The polynomial sub-resultants of P and Q are equal to either 0 or to (up to a constant) polynomials in the remainder sequence of P and Q [1, p.308]. For efficiency, the computations of sub-resultant sequences are usually performed by computing polynomial remainder sequences using some variants of Euclid algorithm.

1.2 Triangular decomposition

First, we project the solutions of S on the x-axis; algebraically, this amounts to computing the resultant of P and Q with respect to the variable y. The roots of that resultant, denoted $Res_y(P,Q) \in \mathbb{Q}[x]$, are the x-coordinates of the solutions of S, and the common roots of their leading coefficients a_p and $b_q \in \mathbb{Q}[x]$. Hence, the resultant of P and Q provides the x-coordinates of the points where the corresponding curves intersect or have a common vertical asymptote. For each such x-coordinate, the difficulty is to compute the y-coordinates of the corresponding intersection points. This can be done with the sub-resultant sequence based on the following fundamental proposition.

Proposition 1 ([3]) *Let $P,Q \in \mathbb{Q}[x,y]$ be two square-free polynomials and consider their sub-resultant sequence with respect to the variable y. Let $\alpha \in \mathbb{R}$ be a root of $Res_y(P,Q)$ such that $a_p(\alpha) \neq 0$ and $b_q(\alpha) \neq 0$. Then*

$$\begin{cases} sres_0(\alpha) = 0, \cdots, sres_{k-1}(\alpha) = 0 \\ sres_k(\alpha) \neq 0 \end{cases}$$
$$\Leftrightarrow Gcd(P(\alpha,y),Q(\alpha,y)) = Sres_k(\alpha,y).$$

We decompose the input system S into a set of triangular systems with respect to the degree of $Gcd(P(\alpha,y),Q(\alpha,y))$ according to Prop. 1. Throughout the decomposition we consider the square-free part of the resultant of P and Q, $Res_y(P,Q) = Sres_0(P,Q)$, and we denote it by $sqrfree(Sres_0(x))$. Taking the square-free part simplifies the computation but it does not preserve the multiplicities of the roots. However, this is actually critical in our algorithm for computing the multiplicities in the fibers needed for the topology computation, as mentioned earlier (Section 2).

According to Prop. 1, we first consider only x-coordinates on which a_p and b_q do not vanish. Geometrically, the roots of a_p correspond to vertical asymptotes of the curve $\mathcal{C}_P$ defined by $P=0$; similarly for b_q and the curve $\mathcal{C}_Q$. We denote by $Fres(x)$ the polynomial $sqrfree(Sres_0(x))$ "without these roots":

$$Fres(x) = \frac{sqrfree(Sres_0(x))}{Gcd(sqrfree(Sres_0(x)), a_p(x)b_q(x))}. \qquad (2)$$

Solutions outside the vertical asymptotes. The solutions of S that do not lie on a vertical asymptote of $\mathcal{C}_P$ or $\mathcal{C}_Q$ have their x-coordinates solutions of $Fres(x)$. The idea is to factorize $Fres(x)$ with respect to the degree of the $Gcd(P(\alpha,y),Q(\alpha,y))$ for α root of $Fres(x)$.

Let $G_1(x) = Gcd(Fres(x), sres_1(x))$. We split $Fres(x)$ in two factors G_1 and $F_1(x) = \frac{Fres(x)}{G_1(x)}$, and we consider the system $S_1 = \begin{cases} F_1(x) \\ Sres_1(x,y) = sres_1(x)y + R(x) \end{cases}$. The roots of S_1 are exactly the roots (α,β) of S such that the degree of the $Gcd(P(\alpha,y),Q(\alpha,y))$ is 1. In other words, α is a root of $Fres(x)$ such that S admits a unique root (α,y) counted with multiplicity.

According to Prop. 1, we split again G_1 with respect, this time, to the polynomial $sres_2(x)$. Let $G_2 = Gcd(G_1, sres_2)$, $F_2 = G_1/G_2$, the system $S_2 = \{F_2(x), Sres_2(x,y)\}$ encodes the solutions (α,β) of S such that the degree of the $Gcd(P(\alpha,y),Q(\alpha,y))$ is 2. In other words, α is a root of $Fres(x)$ such that S admits two roots (α,y) counted with multiplicity.

We go ahead recursively until we decompose completely $Fres(x)$ with respect to the polynomials $sres_1(x), \ldots, sres_q(x)$. At the k-th step the roots of $F_k(x)$ are exactly the roots of $Fres(x)$ verifying $sres_1(x) = 0, \ldots, sres_{k-1}(x) = 0$, and $sres_k(x) \neq 0$.

The result is a set of triangular systems $S_k = \{F_k(x), Sres_k(x,y)\}$ such that $Gcd(F_k(x), sres_k(x)) = 1$.

Solutions on vertical asymptotes. We now consider the solutions of S on an asymptote $x=\alpha$, with α a root of a_p or b_q. Our algorithm differs here from that of [3] in which they detect when such a situation occurs and shear the coordinate system. For sake of brevity, we only detail the case of solutions on a common asymptote, that is $x=\alpha$ with α a root of $D(x)$ the square-free part of $Gcd(a_p,b_q)$. Let P_1 , Q_1 be defined as follows:

$P_1 = \sum_{i=1}^{p}(a_i mod D(x))y^i$, $Q_1 = \sum_{i=1}^{q}(b_i mod D(x))y^i$, where $U(x) \mod D(x)$ is the remainder of the Euclidean division of $U(x)$ by $D(x)$ in $\mathbb{Q}[x]$. It is clear that P_1, Q_1 coincide with P and Q above the roots of $D(x)$, i.e., $\forall \alpha \in \mathbb{R} : D(\alpha) = 0 \Longrightarrow P_1(\alpha,y) = P(\alpha,y)$ and $Q_1(\alpha,y) = Q(\alpha,y)$. We decompose the system $\{P_1, Q_1\}$ as described above but keeping in the resultant only the roots of $D(x)$: $Fres(x) = sqrfree(Gcd(Sres_0(P_1,Q_1), D(x)))$. We thus recover in the resulting systems, solutions located on the common asymptotes.

Output. The output is a set of triangular systems $S_k = \begin{cases} F_k(x) = 0 \\ Sres_k(x,y) = 0 \end{cases}$ such that $Gcd(F_k(x), sres_k(x)) = 1$ and all F_k are square-free. For every k there might be several systems S_k: depending on whether the solution of S_k lies on a vertical asymptote or not, $Sres_k$ denotes the k-th subresultant of P and Q, or of some reductions of P and Q modulo some factors of their leading coefficients a_p and b_q in $\mathbb{Q}[x]$.

1.3 Groebner basis and RUR computation

A specific property of the decomposition obtained above is that $Gcd(F_k(x), sres_k(x)) = 1$. By Bézout identity, there exists $U, V \in \mathbb{Q}[x]$ such that $UF_k + Vsres_k = 1$. Thus, since $Sres_k(x,y) = sres_k(P,Q)y^k + R_k(x,y)$ (Eq. (1)), the system S_k is equivalent to $\tilde{S}_k : \begin{cases} F_k(x) = 0 \\ y^k + \tilde{R}_k(x,y) = 0 \end{cases}$ where $\tilde{R}_k$ is the reduction of VR_k modulo F_k (that is each coefficient is reduced modulo $F_k(x)$); hence $\tilde{R}_k$ has degree in y strictly less than k and strictly less than that of F_k in x. The resulting system thus is a (reduced) lexicographic Gröbner basis [1].

At this stage, we can use a standard RUR algorithm for solving each of these systems [4]. Recall that given a bivariate system, a RUR defines a parameterization of its solutions with four polynomials $f, g_x, g_y, g \in \mathbb{Q}[t]$ such that the solutions of the system are $(\frac{g_x(t_i)}{g(t_i)}, \frac{g_y(t_i)}{g(t_i)})$ for the roots t_i of f. It should be stressed that a RUR preserves the multiplicity of the roots of the system, that is, the multiplicity of a root t_i of f is the multiplicity of the root $(\frac{g_x(t_i)}{g(t_i)}, \frac{g_y(t_i)}{g(t_i)})$ in $\tilde{S}_k$, and thus in S_k. The fact that F_k is square-free yields a geometric interpretation of the multiplicities is the system S_k that will be exploited in our application on the topology of curves.

The particular structure of the systems we obtain yields improvements to the algorithm of Rouillier [4] for computing a RUR. The critical properties of our input systems S_k is that (i) they are Gröbner basis *for the lexicographic order*, which implies that one of the polynomials is univariate, and (ii) there are only two polynomials in the basis. Most of the RUR computations are performed using linear algebra in the quotient algebra $\mathbb{Q}[x,y]/S_k$ (where S_k denote here the ideal associated to the system). Property (ii) implies that a basis of this algebra is simply $\{x^i y^j\}$, for $0 \leqslant i < \text{degree}(F_k)$ and $0 \leqslant j < k$. One important step of the algorithm is to compute the product of all pairs of elements of that basis, reduced modulo the system S_k; this step is substantially simplified by the structure of the basis and by using Property (i). The complexity of this step, and actually of the whole RUR computation, is this way reduced from $O(D^3)$ to $O(D^2)$ where D is the size of the algebra basis.

1.4 Deterministic modular method

The bitsize of the coefficients in the RUR is reasonable in the sense that these coefficients have more or less the same size as those in the resultant [2, §4.2]. However, the size of the coefficients of the lexicographic Gröbner basis in the intermediate computations may be quite large. A standard approach for avoiding such intermediate growth is to use modular computations and the Chinese Remainder Theorem [1]. The difficulty being the design of a deterministic algorithm (rather than a Monte-Carlo algorithm). We show how this approach can be applied here.

Without loss of generality, we can assume that the two input polynomials P and Q have integer coefficients (rather than rational). For brevity, we only describe how to compute the roots outside the vertical asymptotes; the roots on the vertical asymptotes are treated similarly. Let μ be a prime number larger than pq, $\mathbb{Z}_\mu = \frac{\mathbb{Z}}{\mu\mathbb{Z}}$, and let ϕ_μ be the canonical surjection $\mathbb{Z}[x,y] \to \mathbb{Z}_\mu[x,y]$ that transforms the polynomial coefficients modulo μ. We first check that (i) $\phi_\mu(a_p) \neq 0$ and $\phi_\mu(b_q) \neq 0$, then we compute $Fres(P,Q)$ and $Fres(\phi_\mu(P), \phi_\mu(Q))$ (Eq. (2)) and test whether (ii) $\phi_\mu(Fres(P,Q)) = Fres(\phi_\mu(P), \phi_\mu(Q))$ and (iii) $degree(\phi_\mu(Fres(P,Q))) = degree(Fres(P,Q))$. We consider prime numbers in turn until all these properties are satisfied. Then, considering polynomials in $\mathbb{Z}_\mu[x,y]$, we solve the system $\{\phi_\mu(P), \phi_\mu(Q)\}$ as described in Sections 1.2 and 1.3.

Lemma 2 *The sum of the degrees of the systems (the degree of a system being the number of its complex roots counted with multiplicity) in the decomposition applied to $\phi_\mu(P)$ and $\phi_\mu(Q)$ and performed in $\mathbb{Z}_\mu[x,y]$ is greater than or equal to the sum of the degrees of the systems in the decomposition of P and Q performed over the rationals.*

An important remark is that since the RUR preserves multiplicities, the degree of a system is the degree of the first polynomial of its RUR, even when considered over $\mathbb{Z}_\mu[x,y]$, as soon as $\mu > pq$ [4]. If the sum of the degrees of the systems obtained by a computation modulo μ is strictly greater than the sum of the degrees of the systems obtained by the computation over the rationals, we say that μ is an *unlucky prime*. The difficulty is of course to detect unlucky primes.

Our algorithm selects a set of primes μ (as described above), decomposes the system $\{\phi_\mu(P), \phi_\mu(Q)\}$, and determines a RUR for each S_k over $\mathbb{Z}_\mu[x,y]$. If the sums of the degrees of S_k are not the same for all considered primes, the primes for which this sum is not minimal are unlucky, and we discard these primes. Otherwise, either all primes are lucky or they are all unlucky. We then use the Chinese Remainder Theorem to lift the RURs of the systems S_k.

We then check whether the roots of the set of resulting RURs (f, g_x, g_y, g) of each S_k are indeed roots of the system $\{P, Q\}$ with the correct multiplicities. Checking whether the roots are correct is done by substituting the rational coordinates $x = \frac{g_x(t)}{g(t)}$ and $y = \frac{g_y(t)}{g(t)}$ in P and Q and checking that f divides its numerator. A root of multiplicity m corresponding to a root t of f can be verified similarly by substituting the RUR coordinates in the i-th derivatives of P and Q with respect to y (for i from 1 to $m-1$), and checking that t is a root of their gcd with f (more precisely, let $f = \Pi_i f_i^i$ be the decomposition with respect to multiplicities of the first poly of the RUR then t is a root of f_m, and one must check that f_m divides the numerators of the derivatives of P and Q after substitution).

Similarly, the multiplicity m of a root (α, β) of the system (corresponding to a root t of f) can be checked to be correct by substituting the RUR coordinates in the i-th derivatives of P and Q with respect to y (for i from 1 to $m-1$), and checking that t is a root of their gcd with f. Checking that t is not a root of the m-th derivative is unnecessary because m is necessarily larger than or equal to the correct multiplicity of the root (by the proof of Lemma 2).

If, for all S_k, all the roots of the RURs are indeed roots of $\{P, Q\}$, and if their multiplicities are correct, the set of roots (with multiplicities) is necessarily correct by Lemma 2. Otherwise we add some new primes and lift again the result. (We omit here some details, including that a prime may also be unlucky for the computation of the RURs.)

2 Application to the topology of plane curves

We now consider the problem of computing efficiently the topology of a real plane algebraic curve $\mathcal{C}_f$ defined by a bivariate polynomial f in $\mathbb{Q}[x, y]$. Isotop [2] is an algorithm that uses Gröbner basis and RUR computation as a black box for isolating the critical points of $\mathcal{C}_f$, that is the points that are solutions of $\{f, f_y\}$ with $f_y = \frac{\partial f}{\partial y}$. Compared to other algorithms, Isotop is particularly efficient on non-generic curves (i.e., curves with several critical points on some vertical line) because they are treated in the original coordinate system without shearing. However, Isotop is less efficient on some types of *generic* curves, in particular for (i) "random" curves for which the multiplicity of all critical points in their fiber is 2, and (ii) generic curves such that the multiplicity (in the fiber) of some critical points is high. In case (i), the Gröbner basis computation is expensive compared to the sub-resultant sequence decomposition approach of [3] because the decomposition leads to a unique system S_1. In case (ii), the RUR computation (performed as a black box on a Gröbner basis of $\{f, f_y\}$) turns out to also be expensive compared to the decomposition approach because multiplicities in the initial system are kept (refer to [2] for details).

Our contribution is to change the Gröbner basis and RUR black box by the solver presented in section 1. The idea being to decompose the input system when possible while keeping RUR approach for solving these systems. This yields two main changes to the Isotop algorithm: the way to compute critical points, and to determine multiplicities in fibers.

Computing critical points of the curve $\mathcal{C}_f$. Critical points of $\mathcal{C}_f$ are the solutions of the system $S = \{f, f_y\}$. We isolate these solutions by decomposing S in triangular systems S_i and computing their RURs according to the algorithm presented in Sect. 1.

Computing the multiplicities of critical points in their fibers. For a point $\mathbf{p} = (\alpha, \beta)$ on the curve $\mathcal{C}_f$, the **multiplicity of p in its fiber** denoted by $mult(f(\alpha, y), \beta)$ is defined as the multiplicity of β in the univariate polynomial $f(\alpha, y)$. Solving the systems S_i with RURs enables to preserve the multiplicities and, hence, to compute the multiplicities in the fibers even if there are several critical points with the same x-coordinate:

Proposition 3 *If $\mathbf{p}$ is a critical point of $\mathcal{C}_f$ then it is a solution of some system S_i, and its multiplicity in its fiber is its multiplicity is S_i plus one.*

Preliminary experiments. We have computed the ratio of running times of the original Isotop algorithm with its new version, Isotop2, using our solver as explained in Section 2. As expected the ratio is large for generic curves (a ratio between 5 and 14 on curves of degree between 10 and 20) and we also have a significant gain for non-generic curves (a ratio between 1.5 and 15 for degrees between 12 and 28). These preliminary tests hence confirm the theoretical analysis that our approach automatically selects the best strategy in any case.

References

[1] S. Basu, R. Pollack, and M.-R. Roy. *Algorithms in Real Algebraic Geometry*, Springer-Verlag, 2nd edition, 2006.

[2] J. Cheng, S. Lazard, L. Peñaranda, M. Pouget, F. Rouillier, and E. Tsigaridas. On the topology of real algebraic plane curves. *Mathematics in Computer Science*, 4:113–137, 2010. 10.1007/s11786-010-0044-3.

[3] L. González-Vega and I. Necula. Efficient topology determination of implicitly defined algebraic plane curves. *Computer Aided Geometric Design*, 19(9), 2002.

[4] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *J. of Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

Consistent Labeling of Rotating Maps

Andreas Gemsa* Martin Nöllenburg* Ignaz Rutter*

Abstract

Dynamic maps that allow continuous map rotations encounter new issues unseen in static map labeling before. We study the following dynamic map labeling problem: Given a set of points P in the plane with non-overlapping axis-aligned rectangular labels attached to them, the goal is to find a *consistent* labeling of P under rotation that maximizes the number of visible labels for all possible rotation angles. A labeling is called consistent if a single *active* interval of angles is selected for each label such that no two labels intersect at any rotation angle and no point in P is ever occluded by a label.

We introduce a general model for labeling rotating maps and derive basic geometric properties of consistent solutions. We show NP-completeness of the active interval maximization problem and present an efficient polynomial-time approximation scheme.

1 Introduction

Dynamic maps, in which the user can navigate continuously through space, are becoming increasingly important in scientific and commercial GIS applications as well as in personal mapping applications. In particular GPS-equipped mobile devices offer various new possibilities for interactive, location-aware maps. A common principle in dynamic maps is that users can pan, rotate, and zoom the map view—ideally in a continuous fashion. Despite the popularity of several commercial and free applications, relatively little attention has been paid to provably good labeling algorithms for dynamic maps.

Been et al. [2] identified a set of consistency desiderata for dynamic map labeling. Labels should neither "jump" (suddenly change position or size) nor "pop" (appear and disappear more than once) during monotonous map navigation; moreover, the labeling should be a function of the selected map viewport and not depend on the user's navigation history. Previous work on the topic has focused solely on supporting zooming and/or panning of the map [2,3,10], whereas consistent labeling under map rotations has not been considered prior to this paper.

Most maps come with a natural orientation (usually the northern direction facing upward), but applications such as car or pedestrian navigation often rotate the map view dynamically to be always forward facing. Still, the labels should remain horizontally aligned for best readability regardless of the actual rotation angle of the map. A basic requirement in static and dynamic label placement is that labels are pairwise disjoint, i.e., in general not all labels can be placed simultaneously. For labeling point features, it is further required that each label, usually modeled as a rectangle, touches the labeled point on its boundary. It is often not allowed that labels occlude the input point of another label. Figure 1 shows an example of a map that is rotated and labeled. The objective in map labeling is usually to place as many labels as possible. Translating this into the context of rotating maps means that, integrated over one full rotation from 0 to 2π, we want to maximize the number of visible labels. The consistency requirements of Been et al. [2] can immediately be applied for rotating maps.

Figure 1: Input map with five points (a) and two rotated views with some occluded labels (b),(c).

Related Work. Most previous algorithmic research efforts on automated label placement cover *static* labeling models for point, line, or area features. For static point labeling, fixed-position models and slider models have been introduced [4, 7], in which the label, represented by its bounding box, needs to touch the labeled point along its boundary. The label number maximization problem is NP-hard even for the simplest labeling models, whereas there are efficient algorithms for the decision problem whether all points can be labeled in some of the simpler models (see the discussion by Klau and Mutzel [6]. Approximation results [1, 7], heuristics [12], and exact approaches [6] are known for many variants of the static label number maximization problem.

In recent years, *dynamic* map labeling has emerged as a new research topic that gives rise to many unsolved algorithmic problems. Petzold et al. [11] compute a conflict-free labeling for any fixed scale and

*Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT), Germany; {firstname.lastname}@kit.edu

map region from preprocessed conflict information. Mote [9] presents a fast heuristic method for dynamic conflict resolution in label placement that does not require preprocessing. The consistency desiderata of Been et al. [2] for dynamic labeling, however, are not satisfied by either of the methods. Been et al. [3] showed NP-hardness of the label number maximization problem in the consistent labeling model and presented several approximation algorithms for the problem. Nöllenburg et al. [10] recently studied a dynamic version of the alternative boundary labeling model and presented an algorithm that supports continous zooming and panning. None of the existing dynamic map labeling approaches supports map rotation.

2 Model

Let M be a labeled input map, i.e., a set of points $P = \{p_1, \ldots, p_n\}$ in the plane together with a set of pairwise disjoint and axis-aligned rectangular labels $L = \{\ell_1, \ldots, \ell_n\}$, where each point p_i is a point on the boundary $\partial\ell_i$ of its label ℓ_i. We say ℓ_i is *anchored* at p_i. As M is rotated, each label ℓ_i in L remains horizontally aligned and anchored at p_i. Thus, label intersections form and disappear during rotation of M. We take the following alternative perspective. Rather than rotating the points, say clockwise, and keeping the labels fixed we may instead rotate each label around its anchor point counterclockwise and keep the set of points fixed. It is easy to see that both rotations are equivalent.

A *rotation* of L is defined by a rotation angle $\alpha \in [0, 2\pi)$; a *rotation labeling* of M is a function $\phi : L \times [0, 2\pi) \to \{0, 1\}$ such that $\phi(\ell, \alpha) = 1$ if label ℓ is visible or *active* in the rotation of L by α, and $\phi(\ell, \alpha) = 0$ otherwise. We call a labeling ϕ *valid* if for any rotation α the set of labels $L(\alpha) = \{\ell \in L \mid \phi(\ell, \alpha) = 1\}$ consists of pairwise disjoint labels and no label in $L(\alpha)$ contains any point in P (other than its anchor point). We note that a valid labeling is not yet consistent in terms of the definition of Been et al. [2, 3]: having a fixed anchor point labels do not jump and the labeling is independent of the rotation history, but labels may still *pop* during a full rotation from 0 to 2π, i.e., appear and disappear more than once. To avoid this, each label must be active only in a single contiguous range of $[0, 2\pi)$, where ranges are circular ranges modulo 2π so that they may span the input rotation $\alpha = 0$. A valid labeling ϕ, in which for every label ℓ the set $A_\phi(\ell) = \{\alpha \in [0, 2\pi) \mid \phi(\ell, \alpha) = 1\}$ is a contiguous range modulo 2π, is called a *consistent* labeling. For a consistent labeling ϕ the set $A_\phi(\ell)$ is called the *active range* of ℓ. The *length* $|A_\phi(\ell)|$ of an active range $A_\phi(\ell)$ is defined as the length of the circular arc $\{(\cos\alpha, \sin\alpha) \mid \alpha \in A_\phi(\ell)\}$ on the unit circle.

The objective in static map labeling is usually to find a maximum subset of pairwise disjoint labels, i.e., to label as many points as possible. Generalizing this objective to rotating maps means that integrated over all rotations $\alpha \in [0, 2\pi)$ we want to display as many labels as possible. This corresponds to maximizing the sum $\sum_{\ell \in L} |A_\phi(\ell)|$ over all consistent labelings ϕ of M; we call this optimization problem MaxTotal.

3 Properties of consistent labelings

If two labels ℓ and ℓ' intersect under a rotation of α we say they have a (regular) *conflict* at α, i.e., in a consistent labeling at most one of them can be active at α. The set $C(\ell, \ell') = \{\alpha \in [0, 2\pi) \mid \ell \text{ and } \ell' \text{ are in conflict at } \alpha\}$ is called the *conflict set* of ℓ and ℓ'. Since rotations are continuous movements any $C(\ell, \ell') \neq \emptyset$ consists of one or more contiguous and closed angle intervals called *conflict ranges*. Each border of a conflict range corresponds to a rotation angle under which ℓ and ℓ' intersect only on their boundaries. We call these angles *conflict events*.

We show the following lemma in a slightly more general model, in which the anchor point p of a label ℓ can be *any* point within ℓ and not necessarily a point on the boundary $\partial\ell$.

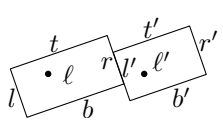
Lemma 1 *For any two labels ℓ and ℓ' with anchor points $p \in \ell$ and $p' \in \ell'$ the set $C(\ell, \ell')$ consists of at most four disjoint contiguous conflict ranges.*

Proof. Assume throughout this proof that without loss of generality p and p' lie on a horizontal line and p is to the left of p'.

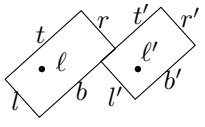
First, we prove that there are at most four disjoint contiguous conflict ranges, and then we show how to compute the conflict events.

The first observation is that due to the simultaneous rotation of all initially axis-parallel labels in L, ℓ and ℓ' remain "parallel" at any rotation angle α. Let l, r, t, b be the left, right, top, and bottom sides of ℓ and let l', r', t', b' be the left, right, top, and bottom sides of ℓ' (defined at a rotation of 0). Since ℓ and ℓ' are parallel, the only possible cases, in which they intersect on their boundary but not in their interior are $t \cap b'$, $b \cap t'$, $l \cap r'$, and $r \cap l'$. Each of these four cases may appear twice, once for each pair of opposite corners contained in the intersection. So there are at most eight conflict events. Figure 2 illustrates two of them. Each conflict event defines a unique rotation angle and hence there are at most four disjoint conflict ranges. This concludes the first part of the proof.

Next we determine the actual rotation angles for which the conflict events occur by considering as an example the intersection of the two sides t and b'. For the label ℓ and its anchor point p let h_t and h_b be the distances from p to t and b. Similarly, let w_l and w_b be the distances from p to l and r (Fig. 3). By h'_t, h'_b,



(a) $r \cap l'$



(b) $b \cap t'$

Figure 2: Two labels ℓ and ℓ' and two of their eight possible boundary intersection events. Anchor points are marked as black dots.

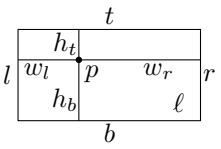


Figure 3: Parameters of label ℓ anchored at p.

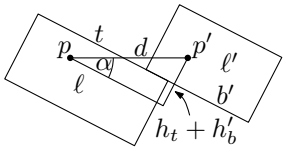


Figure 4: Conflict event for $t \cap b'$.

w'_l, and w'_r we denote the corresponding distances for label ℓ'. Finally, let d be the distance of the two anchor points p and p'. If there is a rotation angle under which t and b' intersect then by simple trigonometric reasoning the two rotation angles at which the conflict events occur are $2\pi - \arcsin((h_t + h'_b)/d)$ (Fig. 4) and $\pi + \arcsin((h_t + h'_b)/d)$. Obviously, we need $d \geq h_t + h'_b$. Furthermore, for the two events induced by $t \cap b'$ to occur we need $d^2 \leq (w_r + w'_l)^2 + (h_t + h'_b)^2$ for the case depicted in Fig. 4 and $d^2 \leq (w_l + w'_r)^2 + (h_t + h'_b)^2$ for the other one. Similar reasoning yields the angles and conditions of the three remaining pairs of conflict events. □

One of the requirements for a valid labeling is that no label may contain a point in P other than its anchor point. For each label ℓ this gives rise to a special class of conflict ranges, called *hard* conflict ranges, in which ℓ may never be active. Obviously, every hard conflict is also a conflict but not vice versa. Similar to conflict events we can define hard conflict events; the union of all conflict and hard conflict events is the set of *label events*. We obtain the hard conflict events immediately from the proof of Lemma 1 by setting the label ℓ' to have height and width 0.

Next we show that the MaxTotal problem can be discretized in the sense that there exists an optimal solution whose active ranges are defined as intervals whose borders are label events. An active range *border* of a label ℓ is an angle α that is characterized by the property that the labeling ϕ is not constant in any ε-neighborhood of α. We call an active range where both borders are label events a *regular* active range.

Lemma 2 *Given a labeled map M there is an optimal rotation labeling of M consisting of only regular active ranges.*

Proof. Let ϕ be an optimal labeling with a minimum number of active range borders that are no label events. Assume that there is at least one active range border β that is no label event. Let α and γ be the two adjacent active range borders of β, i.e., $\alpha < \beta < \gamma$, where α and γ are active range borders, but not necessarily label events. Then let L_l be the set of labels whose active ranges have left border β and let L_r be the set of labels whose active ranges have right border β. For ϕ to be optimal L_l and L_r must have the same cardinality since otherwise we could increase the active ranges of the larger set and decrease the active ranges of the smaller set by some $\varepsilon > 0$ and obtain a better labeling.

We define a new labeling ϕ' that coincides with ϕ except for the labels in L_l and L_r. We set the left border of the active ranges of all labels in L_l and the right border of the active ranges of all labels to γ instead of β. Since $|L_l| = |L_r|$ we shrink and grow an equal number of active ranges by the same amount. Thus the two labelings ϕ and ϕ' have the same objective values but ϕ' has fewer active range borders that are non-label events—a contradiction. □

4 Complexity

By a reduction from planar 3-SAT [8] it can be shown that finding an optimal solution for MaxTotal is NP-hard even if all labels are unit squares and their anchor points are their lower-left corners. Due to space constraints we omit the proof.

Theorem 3 MaxTotal *is NP-complete.*

5 Approximation Scheme

Initially we assume that labels are congruent unit-height rectangles with constant width $w > 1$ and that the anchor points are the lower-left corners of the labels and afterwards explain how to adapt it to more general labeling models. Let $d = \sqrt{w^2 + 1}$.

We present an efficient polynomial-time approximation scheme (EPTAS)[1] that relies on four geometric key properties; they hold for the restricted case of congruent rectangles as well as for the general model of Section 2, where anchor points are arbitrary points within each label or on its boundary, and where the ratio of the smallest and largest width and height, as well as the aspect ratio are bounded by constants: i) the number of anchor points contained in a square is proportional to its area, ii) any label has $O(1)$ conflicts with other labels, iii) any two conflicting labels produce only $O(1)$ conflict regions, and finally, iv) there is an optimal MaxTotal solution where the borders of all active ranges are events.

[1] An EPTAS has a running time of $O(n^c)$ for every fixed ε, where c is constant independent of ε.

Properties (i) and (ii) can be proved by a simple packing argument. Property (iii) follows from property (ii) and Lemma 1. Property (iv) follows immediately from Lemma 2.

Our EPTAS uses the line stabbing technique of Hochbaum and Maass [5]. Consider a grid G where each grid cell is a square with side length $2d$. We can address every grid cell by its row and column index. For any integer k we can remove every k-th row and every k-th column of grid cells, starting at two offsets i and j ($0 \leq i, j \leq k-1$). This yields collections of meta cells of side length $(k-1) \cdot 2d$ that are pairwise separated by a distance of at least $2d$ and thus no two labels whose anchor points lie in different meta cells of the same subset can have a conflict. In total we obtain k^2 such collections of meta cells. We say that a (meta) grid cell c *covers* a label ℓ if the anchor point of ℓ lies inside c.

Determining an optimal solution for the set of labels L_c covered by a meta cell c works as follows. We calculate the set of label events E_c for L_c. Due to Lemma 2 we know that there exists an optimal solution with only regular active ranges defined by events in E_c. Thus, to compute an optimal active range assignment for L_c we can test all possible combinations of regular active ranges for all labels $\ell \in L_c$.

For a given $\varepsilon \in (0, 1)$ we set $k = \lceil 2/\varepsilon \rceil$. Let c be a meta cell for the given k. By a packing argument, we can prove that the number of labels in L_c (and thus also the number of events in E_c) is $O(1/\varepsilon^2)$.

Since we need to test all $O(1/\varepsilon^2)$ possible active ranges for all $O(1/\varepsilon^2)$ labels in L_c we require $O(2^{O(1/\varepsilon^2 \log 1/\varepsilon^2)})$ time to determine an optimal solution for the meta cell c.

With simple arithmetic operations on the coordinates of the anchor points we can determine all non-empty meta cells and store them in a binary search tree. Since we have n anchor points there are at most n non-empty cells in the tree, each of which holds a list of all covered anchor points. Building this data structure requires $O(n \log n)$ time. So for one collection of meta cells the time complexity for finding an optimal solution is $O(n2^{O(1/\varepsilon^2 \log 1/\varepsilon^2)} + n \log n)$. There are k^2 such collections and by the pigeon hole principle the optimal solution for at least one of them is a $(1-\varepsilon)$-approximation.

This yields the result for the simple case of congruent rectangles. In fact we have only used properties (i)–(iv), and there is nothing special about congruent rectangles anchored at their lower-left corners. Hence we can extend the algorithm to the more general labeling model, in which the ratio of the label heights, the ratio of the label widths, and the aspect ratios of all labels are bounded by constants. Furthermore, the anchor points can be any point on the boundary or in the interior of the labels. Finally, we can even ignore the distinction between hard and soft conflicts, i.e., allow that anchor points of non-active labels are occluded. All properties (i)–(iv) still hold. The only change in the EPTAS is to set the width and height of the grid cells to twice the maximum diameter of all labels in L.

Theorem 4 *There exists an efficient polynomial-time approximation scheme that computes a $(1-\varepsilon)$-approximation of* MaxTotal. *Its time complexity is $O((n2^{O(1/\varepsilon^2 \log 1/\varepsilon^2)} + n \log n)/\varepsilon^2)$.*

Acknowledgments. Andreas Gemsa and Martin Nöllenburg received financial support by the *Concept for the Future* of KIT within the framework of the German Excellence Initiative.

References

[1] P. K. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Comput. Geom. Theory Appl.* 11:209–218, 1998.

[2] K. Been, E. Daiches, and C. Yap. Dynamic map labeling. *IEEE Transactions on Visualization and Computer Graphics* 12(5):773–780, 2006.

[3] K. Been, M. Nöllenburg, S.-H. Poon, and A. Wolff. Optimizing active ranges for consistent dynamic map labeling. *Comput. Geom. Theory Appl.* 43(3):312–328, 2010.

[4] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. *Proc. 7th Ann. ACM Sympos. Comput. Geom. (SoCG'91)*, pp. 281–288, 1991.

[5] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM* 32(1):130–136, 1985.

[6] G. W. Klau and P. Mutzel. Optimal labeling of point features in rectangular labeling models. *Mathematical Programming (Series B)* pp. 435–458, 2003.

[7] M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Comput. Geom. Theory Appl.* 13:21–47, 1999.

[8] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.* 11(2):329–343, 1982.

[9] K. D. Mote. Fast point-feature label placement for dynamic visualizations. *Information Visualization* 6(4):249–260, 2007.

[10] M. Nöllenburg, V. Polishchuk, and M. Sysikaski. Dynamic one-sided boundary labeling. *Proc. 18th ACM GIS*, pp. 310–319. ACM Press, November 2010.

[11] I. Petzold, G. Gröger, and L. Plümer. Fast screen map labeling—data-structures and algorithms. *Proc. 23rd Int'l Cartogr. Conf (ICC'03)*, pp. 288–298, 2003.

[12] F. Wagner, A. Wolff, V. Kapoor, and T. Strijk. Three rules suffice for good label placement. *Algorithmica* 30(2):334–349, 2001.

Angular Convergence during Bézier Curve Approximation

J.Li* T.J.Peters† J.A.Roulier‡

February 17, 2011

Abstract

Properties of a parametric curve in $\mathbb{R}^3$ are often determined by analysis of its piecewise linear (PL) approximation. For Bézier curves, there are standard algorithms, known as subdivision, that recursively create PL curves that converge to the curve in distance . The *exterior angles* of PL curves under subdivision are shown to converge to 0 at the rate of $O(\sqrt{\frac{1}{2^i}})$, where i is the number of subdivisions. This angular convergence is useful for determining self-intersections and knot type.

Keywords: Bézier curve, subdivision, piecewise linear approximation, angular convergence.

1 Introduction

A Bézier curve is characterized by an indexed set of points, which form a PL approximation of the curve[1]. Subdivision algorithms recursively generate PL approximations that more closely approximate the curve under Hausdorff distance [8].

Figure 1 shows the first step of a particular subdivision process, known as the de Casteljau algorithm. The initial PL approximation P is used as input to generate local PL approximations, P^1 and P^2. Their union, $P^1 \cup P^2$, is then a PL approximation whose Hausdorff distance is closer to the curve than that of P. In this illustrative example, the construction relies upon generating midpoints of the segments of P, as indicated by providing $\frac{1}{2}$ as an input value to the subdivision algorithm[2].

A brief overview is that subdivision proceeds by first creating the midpoint of each segment of P. Then, these midpoints are connected to create new segments. Recursive creation and connecting of midpoints continues until a final midpoint is created. The union of the segments from the last step then forms a PL approximation. Termination is guaranteed since P has only finitely many segments.

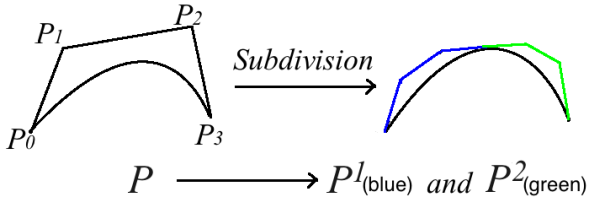


Figure 1: A subdivision

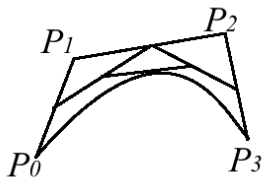


Figure 2: De Casteljau algorithm on P

This subdivision process is purely on PL geometry so that these techniques may be of interest to the computational geometry community. For subdivision, convergence of the PL curves to a Bézier curve under Hausdorff distance is well known [4], but, to the best of our knowledge, the convergence in terms of angular measure has not been previously established. The angular convergence is used to draw conclusions about non-self-intersection of the PL approximation[3]. Non-self-intersection is useful in determining more subtle topological properties, such as unknottedness and isotopic equivalence between a curve and this PL approximant [2]. This isotoptic equivalence has applications in computer graphics, computer animation and scientific visualization and the results presented here were discovered while extending those previous theorems [2]. Further enhancements remain the subject of future research.

For a simple[4] curve, it has been shown that the PL approximation under subdivision will eventually also become simple [6]. The proof relied upon use of the hodo-

*Department of Mathematics, University of Connecticut,Storrs, jili@math.uconn.edu

†Department of Computer Science and Engineering, Univeristy of Connecticut, Storrs, tpeters@cse.uconn.edu

‡Department of Computer Science and Engineering, Univeristy of Connecticut, Storrs, jrou@engr.uconn.edu

[1] In the literature on Bézier curves, this PL approximation is called a *control polygon*. However, to avoid confusion within this community, that terminology will be avoided here.

[2] Other fractional values can be used, but the analysis given here proceeds by reliance upon $\frac{1}{2}$ and midpoints. The details to change from midpoints are not substantive to the analysis presented here.

[3] The authors thank J. Peters for conceptual insights offered during some informal conversatoins about the angular convergence.

[4] A curve is called *simple* if it is non-self-intersecting.

graph[5] and is devoid of the constructive geometric techniques used here. That previous proof did not provide a specific rate of convergence, but the more geometric construction used here easily yields that convergence rate.

2 Definitions and Notation

Exterior angles were defined [5] for closed PL curves, but are adapted here for open curves.

Definition 1 *For an open PL curve with vertices $\{P_0, P_1, \ldots, P_n\}$ in $\mathbb{R}^3$, denote the measures of the exterior angles formed by the oriented line segments to be:*

$$\alpha_1, \ldots, \alpha_{n-1} \text{ satisfying}$$

$$0 \leq \alpha_m \leq \pi \text{ for } 1 \leq m \leq n-1.$$

For example, the exterior angle with measure α_m is formed by $\overrightarrow{P_{m-1}P_m}$ and $\overrightarrow{P_mP_{m+1}}$ and $0 \leq \alpha_m \leq \pi$, as shown in Figure 3). For these open PL curves, it is understood that the exterior angles are not defined at the initial and final vertices.

Definition 2 *Denote $\mathcal{C}(t)$ as the parameterized Bézier curve of degree n with control points $P_m \in \mathbb{R}^3$, defined by*

$$\mathcal{C}(t) = \sum_{m=0}^{n} B_{m,n}(t) P_m, t \in [0,1]$$

where $B_{m,n} = \binom{n}{m} t^m (1-t)^{n-m}$.

Denote the uniform parametrization [7] of the PL curve P by $l(P)_{[0,1]}$ over $[0,1]$, where $P = (P_0, P_1, \cdots, P_n)$. That is:

$$l(P)_{[0,1]}(\frac{j}{n}) = P_j \text{ for } j = 0, 1, \cdots, n$$

and $l(P)_{[0,1]}$ is piecewise linear.

Definition 3 *Discrete derivatives [7] are first defined at the points $t_j's$, where $l(P)_{[0,1]}(t_j) = P_j$ for $j = 0, 1, \cdots, n-1$.*

$$P'_j = l'(P)_{[0,1]}(t_j) = \frac{P_{j+1} - P_j}{t_{j+1} - t_j}$$

Denote $P' = (P'_0, P'_1, \cdots, P'_{n-1})$. Then define the discrete derivative for $l(P)_{[0,1]}$ as:

$$l'(P)_{[0,1]} = l(P')_{[0,1]}$$

Intuitively, the first discrete derivatives are similar to the tangent lines defined for univariate real-valued functions within a standard introductory calculus course.

In order to avoid many annoying techincal considerations and to simplify the exposition, the class of Bézier curves considered will be restricted to those where the derivative never vanishes.

[5]The derivative of a Bézier curve is also expressed as a Bézier curve, known as the *hodograph* [3].

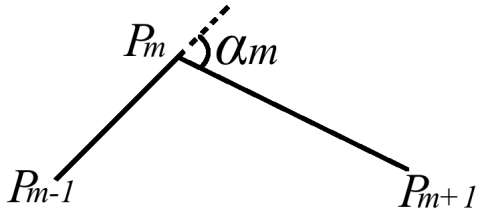


Figure 3: The measure α_m of an exterior angle

Definition 4 *A differentiable curve is said to be regular if its derivative never vanishes.*

Throughout the rest of the presentation, the notation $\mathcal{C}$ will be used for a simple C^1, regular[6] Bézier curve of arbitrary degree n. And i is the number of subdivisions. For convenience, the de Casteljau algorithm is assumed, with a fixed parameter $\frac{1}{2}$.

3 Angular Convergence

For any Bézier curve, after i iterations, the subdivision process generates 2^i open PL curves, whenever $i > 0$ (For $i = 0$, if the original PL curve formed from the control points is closed, then the associated Bézier curve is also closed.) For the i-th subdivision, when $i > 0$, denote each open PL approximation generated as

$$P^k = (P_0^k, P_1^k, \ldots, P_n^k)$$

for $k = 1, 2, 3, \ldots, 2^i$. We consider an arbitrary P^k for the following analysis, where, for simplicity of notation, we repress the superscript and denote this arbitrary curve simply as P, where P has the corresponding parameters of the indicated control points by $t_0, t_1, \cdots, t_n$. And let $l(P,i)$ be the uniform parameterization [7] of P on $[\frac{k-1}{2^i}, \frac{k}{2^i}]$ $k \in \{1, 2, 3, \ldots, 2^i\}$. That is

$$l(P,i) = l(P)_{[\frac{k-1}{2^i}, \frac{k}{2^i}]} \quad and \quad l(P,i)(t_m) = P_m$$

Note from the domain of $l(P,i)$ that

$$t_n - t_0 = \frac{1}{2^i} \tag{1}$$

Furthermore, let

$$\alpha_1, \alpha_2, \cdots, \alpha_{n-1}$$

be the corresponding measures of exterior angles of P (Definition 1).

Consider the Euclidian distance between the discrete derivatives of the two consecutive segments, that is $|l'(P,i)(t_m) - l'(P,i)(t_{m-1})|$, where $|\ |$ denotes Euclidean distance. We will show a rate of $O(\frac{1}{2^i})$ for the convergence

[6]The astute reader will note that some of the development does not require that the curve be regular.

$$|l'(P,i)(t_m) - l'(P,i)(t_{m-1})| \to 0 \text{ as } i \to \infty.$$

This will imply that $cos(\alpha_m) \to 1$ with the same rate and that $\alpha_m \to 0$ at a rate of $O(\sqrt{\frac{1}{2^i}})$. Analogously we may imagine two connected segments in the x-y plane, and if their slopes are close, then their exterior angle is small.

Lemma 1 *For $\mathcal{C}$, the value $|l'(P,i)(t_m) - l'(P,i)(t_{m-1})|$ converges to zero at a rate of $O(\frac{1}{2^i})$.*

Proof.

$$|l'(P,i)(t_m) - l'(P,i)(t_{m-1})|$$
$$\leq |l'(P,i)(t_m) - \mathcal{C}'(t_m)| + |\mathcal{C}'(t_m) - \mathcal{C}'(t_{m-1})| +$$
$$|\mathcal{C}'(t_{m-1}) - l'(P,i)(t_{m-1})|$$

The first and the third terms converge to 0 at a rate of $O(\frac{1}{2^i})$, because the discrete derivative converges to the derivative of the original curve with this rate [7].

Now consider the convergence of the second term. Since $\mathcal{C}'$ satisfies the Lipschitz condition because of it being continuously differentiable, we have

$$|\mathcal{C}'(t_m) - \mathcal{C}'(t_{m-1})| \leq \gamma |t_m - t_{m-1}| \leq \frac{\gamma}{2^i}$$

for some constant γ, where γ does not depend on t_m or t_{m-1}. The second inequality holds by the Equation (1). Therefore $|l'(P,i)(t_m) - l'(P,i)(t_{m-1})|$ converges to zero at a rate of $O(\frac{1}{2^i})$. □

Theorem 2 (Angular convergence) *For $\mathcal{C}$, each exterior angle of the PL curves generated by subdivision converges to 0 at a rate of $O(\sqrt{\frac{1}{2^i}})$.*

Proof. Since $\mathcal{C}(t)$ is assumed to be regular and C^1, the non-zero minimum of $|\mathcal{C}'(t)|$ over the compact set [0,1] is obtained. For brevity, the notations of $u_i = l'(P,i)(t_m)$ and $v_i = l'(P,i)(t_{m-1})$ are introduced. The convergence of u_i to $\mathcal{C}'(t_m)$ [7] implies that $|u_i|$ has a positive lower bound for i sufficiently large, denoted by λ.

Lemma 1 gives that $|u_i - v_i| \to 0$ as $i \to \infty$ at a rate of $O(\frac{1}{2^i})$. This implies: $|u_i| - |v_i| \to 0$ as $i \to \infty$ at a rate of $O(\frac{1}{2^i})$.

Consider the following where the multiplication between vectors is dot product:

$$|cos(\alpha_m) - 1| = |\frac{u_i v_i}{|u_i||v_i|} - 1|$$

$$= |\frac{u_i v_i - v_i v_i + v_i v_i - |u_i||v_i|}{|u_i||v_i|}|$$

$$\leq \frac{|u_i - v_i| + ||v_i| - |u_i||}{|u_i|} \leq \frac{|u_i - v_i| + ||v_i| - |u_i||}{\lambda}$$

It follows from Lemma 1 that the right hand side converges to 0 at a rate of $O(\frac{1}{2^i})$. Consequently by the above inequality $|cos(\alpha_m) - 1| \to 0$ with the same rate.

It follows from the continuity of $arccos$ that α_m converges to 0 as $i \to \infty$.

Taking the power series expansion of cos we get

$$|cos(\alpha_m) - 1| \geq (\alpha_m)^2 \cdot (\frac{1}{2} - |\frac{(\alpha_m)^2}{4!} - \frac{(\alpha_m)^4}{6!} + \cdots|)$$

Considering the expression on the right hand side of the previous inequality, note that for $1 > \alpha_m$,

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots > |\frac{1}{4!} - \frac{(\alpha_m)^2}{6!} + \cdots|.$$

For any $0 < \tau < \frac{1}{2}$, sufficiently many subdivisions will guarantee that α_m is small enough such that $1 > \alpha_m$ and $\tau > (\alpha_m)^2 \cdot e$. Then

$$\tau > (\alpha_m)^2 \cdot e > (\alpha_m)^2 \cdot |\frac{1}{4!} - \frac{(\alpha_m)^2}{6!} + \cdots|.$$

So

$$|cos(\alpha_m) - 1| > (\alpha_m)^2 \cdot (\frac{1}{2} - \tau) > 0.$$

Then convergence of the left hand side implies that α_m converges to 0 at a rate of $O(\sqrt{\frac{1}{2^i}})$. □

4 Non-self-intersections from Subdivision

Even though the original PL approximant might not be simple, if the Bézier curve is simple, then subdivision eventually produces a PL approximant that is simple [6]. So, there must exist some value of i such that, for the i-th subdivision each of the PL curves output as $(P_0^k, P_1^k, \ldots, P_n^k)$ for $k = 1, 2, 3, \ldots, 2^i$ must also be simple. Consistent with the approach taken here, that result will now be shown by a purely PL geometric construction which does not rely upon the hodograph of the Bézier curve.

Lemma 4 is similar to one previously proven [6], where the angles in the previous publication were defined over a different range of values than used here.

The previous definition of exterior angles for open curves (Definition 1) was noted as a specialization of the original use for closed curves, where it was created to unify the concept of total curvature for closed curves that were PL or differentiable.

Definition 5 *The curvature of $\mathcal{C}$ is given by*

$$\kappa(t) = \frac{||\mathcal{C}'(t) \times \mathcal{C}''(t)||}{||\mathcal{C}'(t)||^3}, \quad t \in [0,1]$$

Its total curvature [1] *is the integral: $\int_0^1 |\kappa(t)|\, dt$.*

Definition 6 [5] *For a closed PL curve $\bar{P}$ in $\mathbb{R}^3$, formed from points $P_0, P_1, \ldots, P_n$, its total curvature $\kappa(\bar{P})$ is defined as*

$$\kappa(\bar{P}) = \sum_{m=0}^{n} \alpha_m, .$$

where α_0 and α_m are both defined in the interval $[0, \pi]$, where α_0 is formed from the edges $\overrightarrow{P_n P_0}$ and $\overrightarrow{P_0 P_1}$, while α_n is formed from the edges $\overrightarrow{P_{n-1} P_n}$ and $\overrightarrow{P_n P_0}$

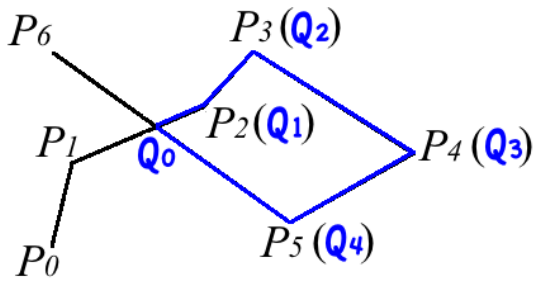


Figure 4: A self-intersecting PL curve

The following Fenchel's Theorem [1] is applicable both to PL curves and to differentiable curves.

Theorem 3 [5] *The total curvature of any closed curve is at least 2π, with equality holding if and only if the curve is convex.*

Lemma 4 *Let $P = (P_0, P_1, \cdots, P_n)$ be an open PL curve in $\mathbb{R}^3$. If $\sum_{j=1}^{n-1} \alpha_j < \pi$, then P is simple.*

Proof. Assume to the contrary that P is self-intersecting. Let k be the lowest index for which the segment $P_{k-1}P_k$ intersects an earlier segment $P_{i-1}P_i$ for $i < k$. Consider the PL curve $P_{i-1} \ldots P_k$ and isolate the single closed loop defined by the intersection as in Figure 4, where the designated intersection point is labeled as Q_0. Denote this single closed loop by $\bar{Q} = (Q_0, Q_1, \cdots, Q_{n'}, Q_0)$, for an appropriately chosen value of n'. Denote the measure of the exterior angle of $\bar{Q}$ at Q_0 by β_0, where $\beta_0 \leq \pi$ (Definition 1). Let κ_l denote the total curvature of this closed loop, where $\kappa_l \geq 2\pi$ (Theorem 3). Then, faithfully index the remaining angles by an oriented traverse of $\bar{Q}$ such that each exterior angle has, respectively, measure β_j for $j = 1, \ldots, n'$. Note that for $j \geq 1$, each β_j is equal to some α_i.

Since $\bar{Q}$ is a portion of P, we have

$$\sum_{i=1}^{n-1} \alpha_j > \sum_{j=1}^{n'} \beta_j.$$

Note also that $\sum_{j=0}^{n'} \beta_j \geq 2\pi$ (Theorem 3). But, then since $\beta_0 \leq \pi$, it follows that $\sum_{j=1}^{n'} \beta_j \geq \pi$ and $\sum_{i=1}^{n-1} \alpha_j \geq \pi$, which is a contradiction.

Two subtleties of the proof are worth mentioning. First, the assumption that each angle $\alpha_i < \pi$ precludes the case of two consecutive edges intersecting at more than a single point. Secondly, the choice of k as the lowest index is sufficiently general to include the case where two non-consecutive edges are coincident. □

Theorem 5 *For $\mathcal{C}$, there exists a sufficiently large value of i, such that after i-many subdivisions, each of the PL curves generated as $P^k = (P_0^k, P_1^k, \ldots, P_n^k)$ for $k = 1, 2, 3, \ldots, 2^i$ will be simple.*

Proof. Since the measure of each exterior angle converges to zero as i increases (Theorem 2) and since each open $P^k = (P_0^k, P_1^k, \ldots, P_n^k)$ has $n-1$ edges, there exists i sufficiently large such that

$$\sum_{j=1}^{n-1} \alpha_j^k < \pi,$$

for each $k = 1, 2, 3, \ldots, 2^i$. Use of Lemma 4 completes the proof. □

5 Conclusions and Future Work

Total curvature is fundamental for determining knot type, as applicable to both PL curves and differentiable curves.

Theorem 6 *Fary-Milnor Theorem* [5]*: If the total curvature of a simple closed curve is less than or equal to 4π, then it is unknotted.*

These local topological properties of the PL approximation for a Bézier curve have been instrumental to showing isotopic equivalence between $\mathcal{C}$ and the polyline approximation generated by subdivision for low degree Bézier curves. Efforts are ongoing to extend that isotopic equivalence to higher degree Bézier curves.

References

[1] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Pretice-Hall, Inc., Upper Saddle River, NJ, 1976.

[2] E.L.F.Moore, T.J.Peters, and J.A.Roulier. Preserving computational topology by subdivision of quadratic and cubic Bézier curves. *Computing, Springer Wien, Special issue on Geometric Modeling*, 79(2-4):317–323, 2007.

[3] G.Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, Inc., San Diego, CA, 1990.

[4] J.Peters and X.Wu. On the optimality of piecewise linear max-norm enclosures based on SLEFES. *International Conference on Curves and Surfaces, Saint-Malo, France*, 2002.

[5] J. W. Milnor. On the total curvature of knots. *Ann.Math.*, 52:248–257, 1950.

[6] M.Neagu, E.Calcoen, and B.Lacolle. Bézier curves: Topological convergence of the control polygon. *6th Int.Conf. on Mathematical Methods for Curves and Surfaces*, pages 347–354, 2000.

[7] G. Morin and R. Goldman. On the smooth convergence of subdivision and degree elevation for Bézier curves. *Computer Aided Geometric Design*, 18:657–666, 2001.

[8] J. Munkres. *Topology*. Prentice Hall, 2nd edition, 1999.

On disjoint crossing families in geometric graphs

Radoslav Fulek*† Andrew Suk*‡

Abstract

A *geometric graph* is a graph drawn in the plane with vertices represented by points and edges as straight-line segments. A geometric graph contains a (k,l)-*crossing family* if there is a pair of edge subsets E_1, E_2 such that $|E_1| = k$ and $|E_2| = l$, the edges in E_1 are pairwise crossing, the edges in E_2 are pairwise crossing, and every edge in E_1 is disjoint to every edge in E_2. We conjecture that for any fixed k, l, every n-vertex geometric graph with no (k,l)-crossing family has at most $c_{k,l}n$ edges, where $c_{k,l}$ is a constant that depends only on k and l. In this note, we show that every n-vertex geometric graph with no (k,k)-crossing family has at most $c_k n \log n$ edges, where c_k is a constant that depends only on k, by proving a more general result which relates extremal function of a geometric graph F with extremal function of two completely disjoint copies of F. We also settle the conjecture for geometric graphs with no $(2,1)$-crossing family. As a direct application, this implies that for any circle graph F on 3 vertices, every n-vertex geometric graph that does not contain a matching whose intersection graph is F has at most $O(n)$ edges.

1 Introduction

A *topological graph* is a graph drawn in the plane with points as vertices and edges as non-self-intersecting arcs connecting its vertices. The arcs are allowed to intersect, but they may not pass through vertices except for their endpoints. Furthermore, the edges are not allowed to have tangencies, i.e., if two edges share an interior point, then they must properly cross at that point. We only consider graphs without parallel edges or self-loops. A topological graph is *simple* if every pair of its edges intersect at most once. If the edges are drawn as straight-line segments, then the graph is *geometric*. Two edges of a topological graph *cross* if their interiors share a point, and are *disjoint* if they do not have a point in common (including their endpoints).

It follows from Euler's Polyhedral Formula that every simple topological graph on n vertices and no crossing edges has at most $3n - 6$ edges. It is also known that every simple topological graph on n vertices with no pair of disjoint edges has at most $O(n)$ edges [10],[7]. Finding the maximum number of edges in a topological (and geometric) graph with a forbidden substructure has been a classic problem in extremal topological graph theory (see [1], [2], [15], [6], [20], [14], [19], [18], [21]). Many of these problems ask for the maximum number of edges in a topological (or geometric) graph whose edge set does not contain a matching that defines a particular intersection graph. Recall that the *intersection graph* of objects $\mathcal{C}$ in the plane is a graph with vertex set $\mathcal{C}$, and two vertices are adjacent if their corresponding objects intersect. Much research has been devoted to understanding the clique and independence number of intersection graphs due to their applications in VLSI design [8], map labeling [3], and elsewhere.

Recently, Ackerman et al. [4] defined a *natural* (k,l)-*grid* to be a set of k pairwise disjoint edges that all cross another set of l pairwise disjoint edges. They conjectured

Conjecture 1 *Given fixed constants $k, l \geq 1$ there exists another constant $c_{k,l}$, such that any geometric graph on n vertices with no natural (k,l)-grid has at most $c_{k,l}n$ edges.*

They were able to show,

Theorem 2 [4] *For fixed k, an n-vertex geometric graph with no natural (k,k)-grid has at most $O(n \log^2 n)$ edges.*

Theorem 3 [4] *An n-vertex geometric graph with no natural $(2,1)$-grid has at most $O(n)$ edges.*

Theorem 4 [4] *An n-vertex simple topological graph with no natural (k,k)-grid has at most $O(n \log^{4k-6} n)$ edges.*

As a *dual* version of the natural (k,l)-grid, we define a (k,l)-*crossing family* to be a pair of edge subsets E_1, E_2 such that

1. $|E_1| = k$ and $|E_2| = l$,
2. the edges in E_1 are pairwise crossing,
3. the edges in E_2 are pairwise crossing,
4. every edge in E_1 is disjoint to every edge in E_2.

*The authors gratefully acknowledge support from the Swiss National Science Foundation Grant No. 200021-125287/1

†EPFL, Lausanne, radoslav.fulek@epfl.ch

‡Courant Institute, New York and EPFL, Lausanne, suk@cims.nyu.edu

We conjecture:

Conjecture 5 *Given fixed constants $k, l \geq 1$ there exists another constant $c_{k,l}$, such that any geometric graph on n vertices with no (k, l)-crossing family has at most $c_{k,l}n$ edges.*

It is not even known if all n-vertex geometric graphs with no k pairwise crossing edges has $O(n)$ edges. The best known bound is due to Valtr [22], who showed that this is at most $O(n \log n)$ for every fixed k. We extend this result to (k, k)-crossing families by proving the following theorem.

Theorem 6 *An n-vertex geometric graph with no (k, k)-crossing family has at most $c_k n \log n$ edges, where c_k is a constant that depends only on k.*

Let F denote a geometric graph. We say that a geometric graph G contains F as a geometric subgraph if G contains a subgraph F' isomorphic to F such that two edges in F' cross if and only if the two corresponding edges cross in F.

We define $ex(F, n)$ to be the extremal function of F, i.e. the maximum number of edges a geometric graph on n vertices can have without containing F as a geometric subgraph. Similarly, we define $ex_L(F, n)$ to be the extremal function of F, if we restrict ourselves to the geometric graphs all of whose edges can be hit by one line.

Let F_2 denote a geometric graph, which consists of two completely disjoint copies of a geometric graph F. We prove Theorem 6 by a straightforward application of the following result.

Theorem 7 $ex(F_2, n) = O((ex_L(F, 2n) + n) \log n + ex(F, n))$

Furthermore, we settle Conjecture 5 in the first nontrivial case.

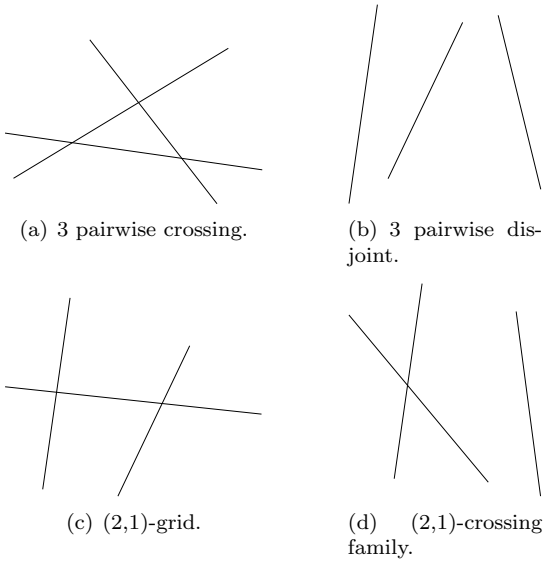
Theorem 8 *An n-vertex geometric graph with no $(2, 1)$-crossing family has at most $O(n)$ edges.*

Note that Conjecture 5 is not true for topological graphs since Pach and Tóth [16] showed that the complete graph can be drawn such that every pair of edges intersect once or twice.

Recall that F is a *circle graph* if F can be represented as the intersection graph of chords on a circle. By combining Theorem 8 with results from [2], [4], and [20], we have the following.

Corollary 9 *For any circle graph F on 3 vertices, every n-vertex geometric graph that does not contain a matching whose intersection graph is F contains at most $O(n)$ edges.*

See Figure 1. We also conjecture the following.



(a) 3 pairwise crossing. (b) 3 pairwise disjoint. (c) (2,1)-grid. (d) (2,1)-crossing family.

Figure 1: Triples of segments corresponding to all circle graphs on three vertices.

Conjecture 10 *For any circle graph F on k vertices, there exists a constant c_k such that every n-vertex geometric graph that does not contain a matching whose intersection graph is F, contains at most $c_k n$ edges.*

As pointed out by Klazar and Marcus [9], it is not hard to modify the proof of the Marcus-Tardos Theorem [19] to show that Conjecture 10 is true when the vertices are in convex position.

For simple topological graphs, we have the following

Theorem 11 *An n-vertex simple topological graph with no $(k, 1)$-crossing family has at most $n(\log n)^{O(\log k)}$ edges.*

2 Relating extremal functions

First, we prove a variant of Theorem 7 when all of the edges in our geometric graph can be hit by a line. As in the introduction let F_2 denote a geometric graph, which consists of two completely disjoint copies of a geometric graph F. We will now show that the extremal function $ex_L(F_2, n)$ is not far from $ex_L(F, n)$.

Theorem 12 $ex_L(F_2,n) \leq O((n+ex_L(F,2n))\log n)$

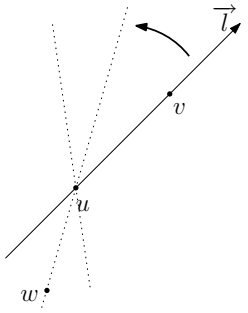
Proof. Let G denote a geometric graph on n vertices that does not contain F_2 as a geometric subgraph, and all the edges of G can be hit by a line. By a standard perturbation argument we can assume that the vertices of G are in general position. As in [5], a *halving edge* uv is a pair of the vertices in G such that the number of vertices on each side of the line through u and v is the same.

Lemma 13 *There exists a directed line $\vec{l}$ such that the number of edges in G that lies completely to the left or right of $\vec{l}$ is at most $2ex_L(F, n/2) + 5n$.*

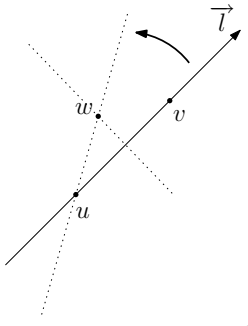
Proof. If n is odd we can discard one vertex of G, thereby loosing at most n edges. Therefore we can assume n is even, and it suffices to show that there exists a directed line $\vec{l}$ such that the number of edges in G that lies completely to the left or right of $\vec{l}$ is at most $2ex_L(F, n/2) + 4n$.

Let uv be a halving edge, and let $\vec{l}$ denote the directed line containing vertices u and v with direction from u to v. Let $e(\vec{l}, L)$ and $e(\vec{l}, R)$ denote the number of edges on the left and right side of $\vec{l}$ respectively. Without loss of generality, we can assume that $e(\vec{l}, L) \leq e(\vec{l}, R)$. We will rotate $\vec{l}$ such that it remains a halving line at the end of each step, until it reaches a position where the number of edges on both sides of $\vec{l}$ is roughly the same.

We start by rotating $\vec{l}$ counterclockwise around u until it meets the next vertex w of G. If initially w lies to the right of $\vec{l}$, then in the next step we will rotate $\vec{l}$ around u (again). See Figure 2(a). Otherwise if w was on the left side of $\vec{l}$, then in the next step we will rotate $\vec{l}$ around vertex w. See Figure 2(b). Clearly after each step in the rotation, there are exactly $n/2$ vertices on each side of $\vec{l}$.



(a) w lies to the right of $\vec{l}$.



(b) w lies to the left of $\vec{l}$.

Figure 2: Halving the vertices of G

After several rotations, $\vec{l}$ will eventually contain points u and v again, with direction from v to u. At this point we have $e(\vec{l}, L) \geq e(\vec{l}, R)$. Since the number of edges on the right side (and left side) changes by at most n after each step in the rotation, at some point in the rotation we must have

$$|e(\vec{l}, L) - e(\vec{l}, R)| \leq 2n.$$

Since G does not contain a F_2 as a geometric subgraph, this implies that

$$e(\vec{l}, L) \leq ex_L(F, n/2) + 2n$$

and

$$e(\vec{l}, R) \leq ex_L(F, n/2) + 2n.$$

Therefore for any n, there exists a directed line $\vec{l}$ such that the number of edges in G that lies completely to the left or right of $\vec{l}$ is at most $2ex_L(F, n/2) + 5n$. □

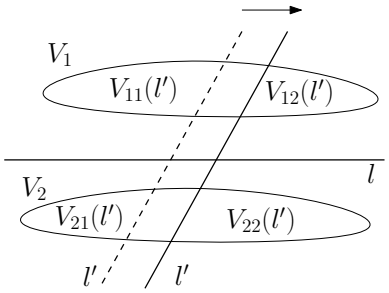


Figure 3: The final partition of the vertex set of G

By Lemma 13 we obtain a line l, which partition the vertices of G into two equal (or almost equal if n is odd) sets V_1 and V_2. Let E' denote the set of edges between V_1 and V_2. By the Ham-Sandwich Cut Theorem [11], there exists a line l' that simultaneously bisects V_1 and V_2. Let $V_{11}(l')$ and $V_{12}(l')$ denote the resulting parts of V_1, and let $V_{21}(l')$ and $V_{22}(l')$ denote the resulting parts of V_2.

Observe that we can translate l' along l into a position where the number of edges in E' that lie completely to the left and completely to the right of l' is roughly the same. In particular, we can translate l' along l such that the number of edges in E' that lies completely to its left or right side is at most $ex_L(F, n) + ex_L(F, n/2+1) + n$ (see Figure 2). Indeed, assume that the number of edges in E' between, say, $V_{12}(l')$ and $V_{22}(l')$ is more than $ex_L(F, n/2 + 1)$. As we translate l' to the right, the number of edges that lie completely to the right of l' changes by at most n as l' crosses a single vertex in G. Therefore we can translate l' into the leftmost position where the number of edges in E' between $V_{12}(l')$ and $V_{22}(l')$ drops below $ex_L(F, n/2 + 1) + n + 1$. Moreover, at this position the number of edges in E' between $V_{11}(l')$ and $V_{21}(l')$ still cannot be more than $ex_L(F, n)$ since G does not contain F_2 as a geometric subgraph.

Thus, all but at most $3ex_L(F, n/2 + 1) + ex_L(F, n) + 6n$ edges of G are the edges between $V_{11}(l')$ and $V_{22}(l')$, and between $V_{12}(l')$ and $V_{21}(l')$. Notice that there exists k, $-1/4 \leq k \leq 1/4$, such that $|V_{11}(l')| + |V_{22}(l')| = n(1/2 + k)$, and $|V_{12}(l')| + |V_{21}(l')| = n(1/2 - k)$. Finally, we are in the position to state the recurrence, whose closed form gives the statement of the theorem:

$ex_L(F_2, n) \leq ex_L(F_2, n(1/2 + k)) + ex_L(F_2, n(1/2 - k)) + 3ex_L(F, n/2 + 1) + ex_L(F, n) + 6n$.

□

Finally, we show how Theorem 12 implies Theorem 7.

Proof. [Proof of Theorem 7.] Let $G = (V, E)$ denote the geometric graph not containing F_2 as a subgraph. Similarly, as in the proof of Lemma 13 we can find a halving line l that hits all but $2ex(F, n/2)+5n$ edges of G. Now, the claim follows by using Theorem 12. □

Theorem 6 follows easily by using Theorem 7 with a result from [22], which states that every n-vertex geometric graph whose edges can be all hit by a line and does not contain k pairwise crossing edges has at most $O(n)$ edges and at most $O(n \log n)$ edges if we do not require a single line to hit all the edges.

References

[1] E. Ackerman, 2006. On the maximum number of edges in topological graphs with no four pairwise crossing edges. In Proceedings of the Twenty-Second Annual Symposium on Computational Geometry (Sedona, Arizona, USA, June 05 - 07, 2006). SCG '06. ACM, New York, NY, 259-263.

[2] P.K. Agarwal, B. Aronov, J. Pach, R. Pollack, and M. Sharir, Quasi-Planar Graphs Have a Linear Number of Edges. In Proceedings of the Symposium on Graph Drawing (September 20 - 22, 1995). F. Brandenburg, Ed. Lecture Notes In Computer Science, vol. 1027. Springer-Verlag, London, 1-7.

[3] P.K. Agarwal, M. van Kreveld, and S. Suri, Label placement by maximum independent set in rectangles, *Comput. Geom. Theory Appl.* **11** (1998), 209–218.

[4] E. Ackerman, J. Fox, J. Pach, and A. Suk, On grids in topological graphs, Proc. 25th ACM Symp. on Computational Geometry (SoCG), University of Aarhus, Denmark, June 2009, 403-412.

[5] T. Dey, Improved Bounds on Planar k-sets and k-levels, Discrete Comput. Geom, 1997, 19, 156–161

[6] J. Fox and J. Pach, Coloring K_k-free intersection graphs of geometric objects in the plane. In Proceedings of the Twenty-Fourth Annual Symposium on Computational Geometry (College Park, MD, USA, June 09 - 11, 2008). SCG '08. ACM, New York, NY, 346-354.

[7] R. Fulek and J. Pach, A computational approach to Conway's thrackle conjecture, Computational Geometry: Theory and Application, 2010, to appear.

[8] D. S. Hochbaum and W. Maass, Approximation schemes for covering and packing problems in image processing and VLSI, *J. ACM* **32** (1985), 130–136.

[9] M. Klazar and A. Marcus,Extensions of the linear bound in the Fredi-Hajnal conjecture, Adv. in Appl. Math. 38 (2006), 258–266.

[10] L. Lovász, J. Pach and M. Szegedy: On Conway's thrackle conjecture, Discrete. Comput. Geom. 18(4) (1997), 369376.

[11] J. Matoušek, Lectures on Discrete Geometry. Springer-Verlag New York, Inc. 2002.

[12] A. Marcus, G. Tardos, Excluded permutation matrices and the Stanley-Wilf conjecture, Journal of Combinatorial Theory Series A, v.107 n.1, p.153-160, July 2004.

[13] J. Pach and J. Solymosi, Crossing patterns of segments. J. Comb. Theory Ser. A 96, 2 (Nov. 2001), 316-325.

[14] J. Pach, R. Radoicic, G. Toth. Relaxing planarity for topological graphs, J. Akiyama, M. Kano (Eds.): Discrete and Computational Geometry, Japanese Conference, JCDCG 2002, Tokyo, Japan, Lecture Notes in Computer Science, 2866, Springer, 2003, 221–232.

[15] J. Pach, F. Shahrokhi, and M. Szegedy. 1994. Applications of the crossing number. In Proceedings of the Tenth Annual Symposium on Computational Geometry (Stony Brook, New York, United States, June 06 - 08, 1994). SCG '94. ACM, New York, NY, 198-202.

[16] J. Pach and G. Tóth, Disjoint edges in topological graphs,in: Combinatorial Geometry and Graph Theory (J. Akiyama et al., eds.),Lecture Notes in Computer Science 3330,Springer-Verlag, Berlin, 2005, 133–140.

[17] J. Pach and G. Tóth, Which crossing number is it anyway?. J. Comb. Theory Ser. B 80, 2 (Nov. 2000), 225-246.

[18] J. Pach, J. Töröcsik, Some Geometric Applications of Dilworth's Theorem. Discrete & Computational Geometry 12: 1-7 (1994)

[19] G. Tardos, G. Tóth: Crossing stars in topological graphs, Japan Conference on Discrete and Computational Geometry 2004, Lecture Notes in Computer Science 3742 Springer-Verlag, Berlin, 184-197.

[20] G. Tóth, 2000. Note on geometric graphs. J. Comb. Theory Ser. A 89, 1 (Jan. 2000), 126-132.

[21] G. Tóth and P. Valtr. 1998 Geometric Graphs with Few Disjoint Edges. Technical Report. UMI Order Number: 98-22., Center for Discrete Mathematics & Theoretical Computer Science.

[22] P. Valtr, Graph drawings with no k pairwise crossing edges, in: G. Di Battista (ed.), Graph Drawing '97, Rome, September 1997, pp. 205-218

Long Monotone Paths in Convex Subdivisions

Günter Rote*

Abstract

Consider a connected subdivision of the plane into n convex regions where every vertex has degree at most d. Then, for every vertex there is a path with at least $\Omega(\log_d n)$ edges through this vertex that is monotone in some direction. This bound is best possible.

1 Introduction

Definition 1 *A* direction *is given by a unit vector e. A path $x(t)$, $t \in I$ for some interval I, is* monotone in direction e *if the inner product $x(t) \cdot e$ is a strictly increasing function of t. A path is* weakly monotone in direction e *if the inner product is a nondecreasing function of t.*

We say that a path is (weakly) monotone *if it is (weakly) monotone in some direction.*

We are looking for long monotone paths in planar subdivisions. A subdivision of the plane into n convex regions has $O(n)$ vertices and $O(n)$ straight edges, including some infinite rays. When we speak about monotone paths, we mean paths along the vertices and edges of this graph. To exclude some trivial cases, we assume that the edges form a connected graph (and we say that the subdivision is *connected*).

Theorem 1 *Let P be a connected subdivision of the plane into n convex faces in which every vertex has degree at most d. Then, for every vertex v, there is a weakly monotone path with at least $\Omega(\log_d n)$ edges starting at v.*

Related Results. If P is the vertical projection of a piecewise linear convex terrain $\hat{P}$, one can apply a polarity transform to this terrain with respect to the paraboloid $z = x^2 + y^2$, yielding another piecewise linear convex terrain $\hat{P}^*$. The projection P^* of this terrain is a *reciprocal diagram* of P (cf. [1]): its graph is dual to the graph of P, in the sense that vertices of P^* correspond to faces of P and vice versa. Each edge in P^* has a corresponding edge in P, and moreover, these two edges are perpendicular. (This last property distinguishes a reciprocal diagram from a general drawing of the dual graph of P.)

*Institut für Informatik, Freie Universität Berlin, Takustraße 9, 14195 Berlin, Germany, rote@inf.fu-berlin.de

A monotone path in P becomes a *monotone face sequence* in P^*: in such a sequence, one can go from a face A to an adjacent face B whenever there is a ray in the specified direction e which crosses the common edge of A and B in the direction from A to B.

Monotone face sequences were studied by Chazelle, Edelsbrunner and Guibas [2]. They established the following bound:

Theorem 2 *Let Q be a subdivision of the plane into n convex faces in which every face is adjacent to at most d neighboring faces. Then, there is a monotone face sequence of length at least*

$$\Omega(\log_d n + \log n / \log\log n).$$

Chazelle et al. showed that a monotone sequence of this length can even be achieved by the faces intersected by a line. Moreover, they showed that the bound is tight, by giving families of subdivisions that have no long face sequences.

Thus, for subdivisions P that are projections of a convex terrain, the question about long monotone paths is completely answered the applying above theorem for $Q = P^*$.

However, for general subdivisions, the problems are not directly related, and in fact, the answers are different: We will see in Section 3 that one cannot add a term $\Omega(\log n / \log\log n)$ (or any other term that grows to infinity) to the bound of Theorem 1.

Motivation. We were led to the question of Theorem 2 by the complexity analysis of an algorithm for partial matching between two finite planar point sets under translations [4]. There, we could show that a certain subdivision Q contained no monotone face sequence longer than some bound X which is polynomial in the input parameters. If Theorem 2 were true with some stronger lower bound of the form $\Omega(n^\alpha)$ for $\alpha > 0$, this would have implied a polynomial bound on the number of faces of Q.

2 Proof of Theorem 1

A basic observation for convex subdivisions is that for any vertex v and any given generic direction e, there is always an outgoing edge from v that is weakly monotone in direction e. Theorem 1 follows quite straightforwardly from the following lemma:

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

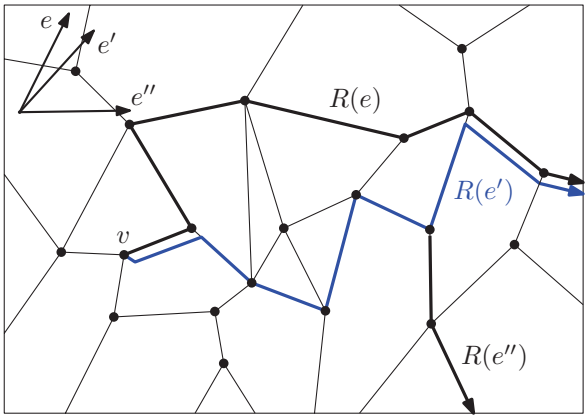


Figure 1: The rightmost path $R(e)$ starting from vertex v in direction e

Lemma 3 *Let v be a vertex in a connected convex planar subdivision P. Then there is a spanning tree rooted a v and containing all infinite rays, such that all paths starting at v are weakly monotone.*

A wall of bricks shows that the lemma does not hold with (strictly) monotone paths. However, if there are no angles of 180°, the statement extends to strictly monotone paths.

Proof. For a generic direction e we can define the rightmost path $R(e)$ starting at v as follows, see Figure 1: We start at v and always follow the rightmost outgoing edge that is weakly monotone in direction e until we arrive at an unbounded ray.

Now we start rotating e clockwise. At some direction e', $R(e')$ will be different from $R(e)$. $R(e)$ is still weakly monotone in direction e'. Now, any vertex u (and any edge) in the region between $R(e)$ and $R(e')$ can be reached by a monotone path in direction e': We simply start at u and go monotonically in the direction opposite to e' until we hit $R(e)$ or $R(e')$. From there, we follow $R(e)$ or $R(e')$ to v. In this way, we can form a spanning tree of all vertices and all infinite rays between $R(e)$ and $R(e')$ with the desired properties.

Continuing the rotation in this way, we eventually reach all vertices and all infinite rays. □

3 Upper Bounds

If the maximum degree d is $\Omega(n)$, Theorem 1 gives only a trivial statement. The example of Figure 2 shows that, indeed, there is no non-constant lower bound on the length of monotone paths in this case, even for a triangulation.

The graph has $2k$ vertices arranged on two concentric rings around a central vertex and connected in a zigzag manner. Each vertex is connected to the center and has also an infinite ray outwards. The radius

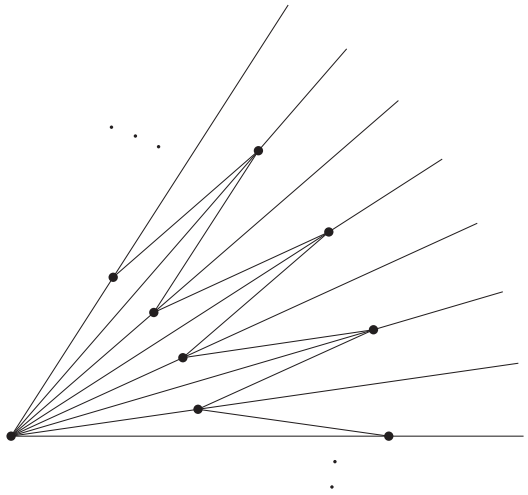


Figure 2: A subdivision with only constant-size monotone paths

of the outer ring is (at least) twice as large as the radius of the inner ring. If we disregard the central vertex and the infinite edges, the zigzag remains. One can easily see that a monotone path can use at most three successive edges of the zigzag. Since a path can go through the center at most once and can use at most two infinite rays, a constant upper bound follows. In fact, it is easy to show that a monotone path can have at most eight edges.

4 Open Question

The example of Figure 2 has $\Omega(n)$ infinite rays. Are there similar examples with constantly many rays?

Acknowledgments

This work was done while I was a guest of CIB (Centre Interfacultaire Bernoulli) in Lausanne. I gratefully acknowledge the support of the CIB and the NSF (National Science Foundation). I thank János Pach for insistingly asking the question to which Theorem 1 gives the answer. I thank the organizers for the tasty chocolate.

References

[1] F. Aurenhammer. A criterion for the affine equality of cell complexes in R^d and convex polyhedra in R^{d+1}. *Discrete Comput. Geom.*, 2:49–64, 1987.

[2] B. Chazelle, H. Edelsbrunner, and L. J. Guibas. The complexity of cutting complexes. *Discrete Comput. Geom.*, 4:139–181, 1989.

[3] D. Grace. All theories proven with one graph. *The Journal of Irreproducible Results*, 50(1):1, 2006.

[4] G. Rote. Partial least-squares point matching under translations. In J. Vahrenhold, editor, *26th European Workshop on Computational Geometry (EuroCG'10)*, pages 249–251, Mar. 2010.

Computing Strongly Homotopic Line Simplification in the Plane

Shervin Daneshpajouh* Mohammad Ali Abam† Lasse Deleuran‡ Mohammad Ghodsi§

Abstract

We study a variant of the line-simplification problem where we are given a polygonal path $P = p_1, p_2, \ldots, p_n$ and a set S of m point obstacles in a plane, and the goal is to find the optimal strongly-homotopic simplification, that is, a minimum subsequence $Q = q_1, q_2, \ldots, q_k$ ($q_1 = p_1$ and $q_k = p_n$) of P defining a polygonal path which approximates P within a given error ε and every link $q_l q_{l+1}$ corresponding to the shortcut $p_i p_j$ is homotopic to the sub-path $p_i, \ldots, p_j$ of P. We present a general method running in time $O(n(m+n)\log(nm))$ for identifying every shortcut $p_i p_j$ that is homotopic to the sub-path $p_i, \ldots, p_j$ of P, called a homotopic shortcut. In the case of x-monotone paths we propose an efficient and simple method to compute all homotopic shortcuts in $O(m\log(nm) + n\log n\log(nm) + k)$ time where k is the number of homotopic shortcuts.

1 Introduction

Introduction

Suppose we want to visualize a large geographical map as a collection of non-intersecting chains representing some features like rivers or country borders, and points representing places like cities, at different levels of details. This leads us to simplify the map. A simplified map must resemble the original map, that is, it must satisfy conditions (i) the distance between each point on a original chain and the simplified chain is within a given error tolerance, and (ii) each original chain and the simplified chain must be in the same homotopy class; roughly speaking it means that if a point (city for instance) is below a chain (river for example), the point must remain below the simplified chain. We however, pay our attention to a simpler variant of the above problem where we are given a polygonal path $P = p_1, p_2, \ldots, p_n$ and a set S of m point obstacles in a plane, and we wish to simplify the path P such that the simplified path is close enough to stay within the given error tolerance of the original path, while still homotopic to the original path.

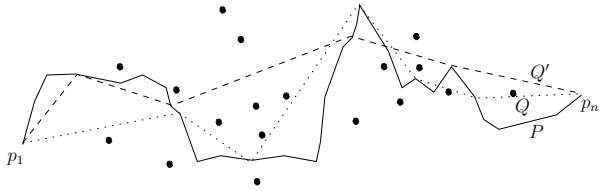


Figure 1: Two simplifications Q and Q' of path P. Only Q is strongly homotopic to P.

In this paper we study the restricted version of the line-simplification, in which the vertices of Q are a subsequence of P and each $q_l q_{l+1}$ is called a *link*. Each link $q_l q_{l+1}$ of the simplified path corresponds to a shortcut $p_i p_j$ (with $j > i$) of the original path P, and the error of the link is defined as the distance between $p_i p_j$ and the sub-path $p_i, \ldots, p_j$ denoted by $P(i,j)$. To measure the distance between $p_i p_j$ and $P(i,j)$ the Hausdorff distance and the Fréchet distance are often used. The *error* of the simplification Q denoted by $\text{error}(Q,P)$ is now defined as the maximum error of any of its links. A simplification Q is homotopic to path P if it is continuously deformable to P without passing over any points of S while keeping its end-vertices fixed. A path P and its simplification Q are *strongly homotopic* if for any link $q_l q_{l+1}$ of Q corresponding to the shortcut $p_i p_j$, the sub-path $P(i,j)$ and $p_i p_j$ are homotopic. For ease of presentation, such a shortcut is called a *homotopic shortcut*. Fig.1 illustrates two simplifications Q and Q'. The simplification Q is strongly homotopic to P but Q' is not; so only Q is of our interest. If a path P and its simplification Q are strongly homotopic, they are clearly homotopic as well but if P and Q are homotopic, they are not necessarily strongly homotopic. However, if P is x-monotone, we can simply conclude P and Q are strongly-homotopic if they are homotopic.

Related work. De Berg *et al.*[2, 3] were first to study homotopic line simplification in the restricted model for the Hausdorff distance under the Euclidean metric. Their algorithm running in $O(n(n+m)\log n)$ time finds the optimal simplification provided that P is x-monotone; otherwise the simplification is not guaranteed to be optimal. Guibas *et al.*[8] showed that the optimal homotopic line-simplification prob-

*Computer Engineering Department, Sharif University of Technology, daneshpajouh@ce.sharif.edu

†Technische Universität Dortmund, mohammad-ali.abam@tu-dortmund.de

‡MADALGO, Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation. Computer Science Department, Aarhus University, ld@madalgo.au.dk

§Computer Engineering Department, Sharif University of Technology, ghodsi@sharif.edu

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

lem in the unrestricted model is NP-hard when the simplification is forced to be non-self-intersecting like the original one. A general version, subdivision simplification, was later proved to be MIN PB-complete, that is even hard to be approximated, and some heuristic algorithms were given by Estkowski and Mitchell [7].

Our results. We propose general methods to efficiently compute all homotopic shortcuts which combined with the Imai and Iri's general framework, give us the optimal strongly-homotopic simplification. For x-monotone paths P of length n, our first algorithm computes all homotopic shortcuts in $O(m\log(nm) + n\log n\log(nm) + k)$ time where k is the number of homotopic shortcuts. In the case of general paths P of length n, we propose an $O(n(m+n)\log(nm))$-time algorithm to compute all homotopic shortcuts.

2 Optimal strongly-homotopic simplification

Our approach, like almost all algorithms given in the restricted model, is based on Imai and Iri's general framework in which for a given error ε, an unweighted directed graph $G_\varepsilon(P)$ (or simply G_ε) is defined as follows: $G_\varepsilon = (V, E_\varepsilon)$ where $V = \{p_1, \ldots, p_n\}$ and $E_\varepsilon = \{(p_i, p_j) | d_F(p_ip_j, P(i,j)) \leqslant \varepsilon\}$ where $d_F(p_ip_j, P(i,j))$ is the F distance of shortcut p_ip_j and $P(i,j)$, for some error function F. Each simplification Q with the error of at most ε corresponds to a path in G_ε from p_1 to p_n and therefore, the optimal simplification is the shortest path in G_ε from p_1 to p_n in time $O(|G_\varepsilon|)$. Our general algorithm is as follows. We first compute G_ε in the absence of the obstacles, and then test whether each edge p_ip_j in G_ε is homotopic to $P(i,j)$ in the presence of the obstacles. We remove non-homotopic shortcuts from G_ε and finally compute the shortest path in G_ε from p_1 to p_n by a breath first search to obtain the optimal strongly-homotopic simplification. The main step of our algorithm is how to compute homotopic shortcuts; the rest exists in the literature as already mentioned. This step can be done in $O((n^3 + n^2m)\log(nm))$ time by individually applying the homotopy-testing algorithm given in [4] to each edge in G_ε and its corresponding sub-path. This of course is far from being efficient. Therefore, the challenging question is whether it is possible to efficiently compute all homotopic shortcuts. We affirmatively answer this question using some novel ideas and nice observations and obtain the following results using Lemmas 3 and 6.

Theorem 1 *Suppose P is a non-self-intersecting polygonal path with size n in a plane containing m point obstacles. Suppose that $T_F(n)$ is the time needed to compute G_ε under the error function F in the absence of the obstacles. Therefore (i) if P is x-monotone, the optimal strongly-homotopic simplification can be computed in $T_F(n) + O(m\log(nm) + n\log n\log(nm)+k)$ time where k is the number of homotopic shortcuts. (ii) if P is a general path, the optimal strongly-homotopic simplification can be computed in $T_F(n) + O(n(m+n)\log(nm))$ time.*

2.1 Computing homotopic shortcuts for x-monotone paths

The main idea behind our x-monotone algorithm is to enclose P inside a simple polygon $\Psi(P,S)$ such that p_ip_j is a homotopic shortcut if and only if p_i can see p_j inside the simple polygon $\Psi(P,S)$.

We define the simple polygon $\Psi(P,S)$ as follows. We first determine the aboveness relation of the obstacles and path P in $O(m\log n)$ time by locating each obstacle with respect to the path P in $O(\log n)$ time. We then compute in $O(n+m)$ time an axis-parallel rectangle containing the path vertices and obstacles in its interior. Let λ be a sufficiently small number less than the x-coordinate difference of any two points of the path vertices and obstacles—λ can simply be computed in $O((n+m)\log(nm))$ time. For each obstacle s_i above P, we define two points s_i^+ and s_i^- on the upper boundary of the rectangle with x-coordinates $x(s_i) + \lambda$ and $x(s_i) - \lambda$ respectively where $x(s_i)$ is the x-coordinate of s_i. We connect s_i to s_i^+ and s_i^- and remove the part of the rectangle joining s_i^+ to s_i^-. Similarly, we define s_i^+ and s_i^- and the associated edges for obstacles s_i below P. All together, this gives us the simple polygon $\Psi(P,S)$.

Lemma 2 *A shortcut p_ip_j is a homotopic shortcut if and only if p_i can see p_j inside $\Psi(P,S)$.*

The above lemma reduces the problem to computing the visibility graph of n points inside a simple polygon with size $3m+4$. Thanks to Ben-Moshe *et al.*[1] who presented an $O(m + n\log n\log(mn) + k)$-time algorithm to compute all visible pairs in $O(n + m + k)$ space, where k is the number of visible pairs. Putting it all together we get the following result:

Lemma 3 *Let $P = p_1, \ldots, p_n$ be an x-monotone polygonal path and let S be a set of m point obstacles in a plane. All homotopic shortcuts p_ip_j can be computed in $O(m\log(nm) + n\log n\log(nm) + k)$ time and $O(n+m+k)$ space where k is the number of homotopic shortcuts.*

2.2 Computing homotopic shortcuts for general paths

In this section we describe an algorithm to compute all homotopic shortcuts p_ip_j in $O(n(m+n)\log(nm))$ time. We fix $i = 1$ and show all homotopic shortcuts p_1p_j can be together computed in $O((m+n)\log(nm))$

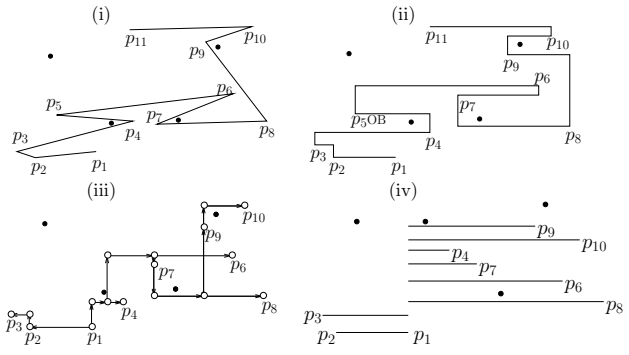


Figure 2: (i) The original path $P = p_1, \ldots, p_{11}$. (ii) The rectified path. (iii) The tree $\mathcal{T}$. The nodes of $\mathcal{T}$ are denoted by empty circles. (iv) The rectified shortcuts.

time which trivially can be extended to any i obtaining our main result as i changes from 1 to n.

One way of representing a path α is to write the sequence of obstacles that it passes above (overbar) and below (underbar). An adjacent pair of repeated obstacles can be deleted by deforming the path without changing the homotopy class. This deletion can be continued until a *canonical sequence* is obtained. A repeated obstacle with opposite bars is called a turn obstacle of α. The path α can be represented in the rectified form, so-called *rectified path*, where each maximal x-monotone sub-path is treated as a horizontal segment whose y-coordinate is its rank in the total order. We define the *canonical rectified path* (denoted by CRP(α)) to be the shortened form of the rectified path α which represents the canonical sequence of α.

Rectifying P. Since P is non-self-intersecting, the edges of P and the obstacles of S induce an aboveness relation which is acyclic and computable in time $O((n+m)\log(nm))$ [9]. The aboveness relation indeed defines a partial order. This partial order can be easily extended to a total order. Let $\text{rank}_p(\mathcal{O})$ be the rank of an object $\mathcal{O}$ (obstacle or edge) in the total order. By letting the y-coordinate of any point of object $\mathcal{O}$ be $\text{rank}_p(\mathcal{O})$, the path P gets rectified, i.e, each edge is represented as a horizontal segment. We join two horizontal segments corresponding to two consecutive edges in P by a vertical segment in order to maintain the original connectivity—see Fig. 2(ii).

Orthogonal-range queries. Our algorithm relies on a three-sided orthogonal range-query data structure defined over m point obstacles in order to compute the the *canonical rectified path*, denoted by CRP($P(1,i)$).This data structure is used to find the closest obstacle to any given vertical segment. We use Chazelle's data structure [5] that preprocesses the obstacles in time $O(m \log m)$ while using $O(m)$ space in order to answer three-sided orthogonal-range queries in $O(\log m)$ time.

Canonical rectified paths. After rectifying P and constructing the orthogonal-range-query data structure we are ready to compute all CRP(i) inductively, i.e, CRP($i+1$) is obtained from CRP(i). Since separately maintaining CRP(i) may require $O(n^2)$ space, we implicitly store them in a tree $\mathcal{T}$ (See Fig. 2(iii)) such that CRP(i) can be extracted by traversing $\mathcal{T}$ from the root to either a leaf or an internal node of $\mathcal{T}$, provided that it is x-monotone and it is known that if CRP(i) is not x-monotone, it can not be homotopic with its corresponding shortcut p_1p_i. Due to lack of space we omit the construction of tree $\mathcal{T}$ from this version but it can be found in the full version[6]. To this end we state the final result in the following lemma.

Lemma 4 *All x-monotone* CRP(i) *can be encoded into a tree $\mathcal{T}$ of size $O(n)$ where each x-monotone* CRP(i) *corresponds to a path from the root to a node or leaf in the tree $\mathcal{T}$.*

Rectifying shortcuts. Let $\mathcal{I}$ be the set of indices i such that CRP(i) is x-monotone. Indeed $\mathcal{I}$ presents all shortcuts remaining candidates to be homotopic shortcuts after computing all CRP(i). Consider all shortcuts p_1p_i where $i \in \mathcal{I}$ and m obstacles in S. They induce an aboveness relation defining a partial order which can be simply extended to a total order. Let $\text{rank}_s(\mathcal{O})$ be the rank of an object $\mathcal{O}$ (obstacle or shortcut) in this total order. We set the y-coordinate of any point of object $\mathcal{O}$ to be $\text{rank}_s(\mathcal{O})$. This rectifies all shortcuts. Note that there is not any relation between $\text{rank}_p(s_i)$ and $\text{rank}_s(s_i)$ for obstacle s_i as they are coming from two different total orders.

Testing homotopy for tree $\mathcal{T}$ and the rectified shortcuts. To summarize, we have two planes: (i) the plane $\mathcal{H}_1$ includes the obstacles and tree $\mathcal{T}$, and (ii) the plane $\mathcal{H}_2$ includes the obstacles and the rectified shortcuts corresponding to x-monotone CRP(i) encoded in $\mathcal{T}$. The tree $\mathcal{T}$ has $O(n)$ edges and all its edges embedded in $\mathcal{H}_1$ are either horizontal or vertical—note that $\mathcal{T}$ is not necessary non-self-intersecting and therefore a partial order of paths encoded in $\mathcal{T}$ may not exist. The y-coordinates of all horizontal segments and the obstacles in $\mathcal{H}_1$ come from rank_p while in $\mathcal{H}_2$ come from rank_s.To this end, we recall that $\mathcal{I}$ is the set of indices i such that CRP(i) is x-monotone. For a horizontal edge e of $\mathcal{T}$, let above(e) (below(e)) be the set of obstacles s_j satisfying (i) $\text{rank}_p(s_j) > \text{rank}_p(e)$ ($\text{rank}_p(s_j) < \text{rank}_p(e)$) and (ii) the x-coordinate of s_j lies between the x-coordinates of the endpoints of e.

It is easy to see the following lemma.

Lemma 5 *For any $i \in \mathcal{I}$, the rectified shortcut p_1p_i and* CRP(i) *are homotopic if and only if for any horizontal edge e of $\mathcal{T}$ appearing on* CRP(i) *and any obstacle $s_j \in$ above(e) ($s_j \in$ below(e)), the condition (A) $\text{rank}_s(p_1p_i) < \text{rank}_s(s_j)$ ($\text{rank}_s(p_1p_i) > \text{rank}_s(s_j)$) is satisfied.*

This lemma simply implies that any edge e and any object $s_j \in$ above(e) ($s_j \in$ below(e)) violating the condition (A) removes p_1p_i of being a homotopic shortcut and if there is not such a pair e and s_j, the shortcut p_1p_i is definitely a homotopic shortcut. However, testing condition (A) for each edge e and any $s_j \in$ above(e) is costly. The following useful observation shows among all obstacles either above or below e, at most two obstacles together with e are necessary to be tested whether they satisfy condition (A).

Observation 1 *It suffices to verify the condition (A) for each horizontal edge e of $\mathcal{T}$ and the obstacle s_j which has the minimum (maximum)* rank_s *in* above(e) (below(e))*; let* $\min(\text{above}(e)) = \text{rank}_s(s_j)$ ($\max(\text{below}(e)) = \text{rank}_s(s_j)$).

Now, the main question is how fast we can perform the verification of condition (A) for any horizontal edge e of $\mathcal{T}$ to remove non-homotopic shortcuts as we know above(e) or below(e) may contain many obstacles and $\mathcal{I}(e)$ may contain several indices where $\mathcal{I}(e)$ is the set of all indices $i \in \mathcal{I}$ that e appears in CRP(i). From now on, we pay our attention to above(e) as below(e) can be handled similarly.

In order to collectively verify the condition (A) in which an edge e is involved, we define a new ordering of elements in $\mathcal{I}$ such that elements in $\mathcal{I}(e)$ become consecutive. This ordering is obtained by an in-order traversal of $\mathcal{T}$—note that for any $i \in \mathcal{I}$, there is a node in $\mathcal{T}$ labeled with p_i. Let $\sigma(i)$ be the rank of $i \in \mathcal{I}$ in this ordering. With each $i \in \mathcal{I}$, we associate the point $(\sigma(i), \text{rank}_s(p_1p_i))$ in a new plane $\mathcal{H}$. Each edge e and its associated $\min(\text{above}(e))$ define a three-sided orthogonal range in $\mathcal{H}$. It is easy to see every $i \in \mathcal{I}$ whose corresponding point lies in this range, violates the condition (A) and consequently must be removed from $\mathcal{I}$. As these three-sided ranges are available in advance, we can sweep points and ranges in $\mathcal{H}$ from top to bottom and maintain a binary tree over the points based on their x-coordinates (i.e. their σ-coordinate). Upon processing a three-sided range, we simply remove $O(\log n)$ subtrees from the binary tree. Therefore the sweeping takes $O(n \log n)$ time in total. After handling below(e) for horizontal edges e of $\mathcal{T}$ in a similar manner, any remaining shortcut is a homotopic shortcut. Putting it all this together we get the following result.

Lemma 6 *Let $P = p_1, \ldots, p_n$ be a non-self-intersecting polygonal path and let S be a set of m point obstacles in a plane. All homotopic shortcuts p_ip_j can be computed in $O(n(m+n)\log(nm))$ time and $O(n+m+k)$ space where k is the number of homotopic shortcuts.*

3 Conclusion

We have proposed algorithms to compute an optimal line simplification Q (in the restricted model) of a polygonal path P with size n being strongly-homotopic to P with respect to m point obstacles in a plane. For non-self-intersecting x-monotone and general path P, our algorithms run in $T_F(n) + O(m\log(nm) + n\log n\log(nm) + k)$ and $T_F(n) + O(n(m+n)\log(nm))$ time respectively where k is the number of homotopic shortcuts and $T_F(n)$ is the time needed to compute G_ε under the error function F .

References

[1] Ben-Moshe, B., Hall-Holt, O., Katz, M., Mitchell, J.: Computing the visibility graph of points within a polygon. *In Proc. Annual Symposium on Computational Geometry*, 27–35, 2004.

[2] de Berg, M., van Kreveld, M., Schirra, S.: A new approach to subdivision simplification. *International Symposium on Computer Assisted Cartography*, 04, 79–88, 1995.

[3] de Berg, M., van Kreveld, M., Schirra, S.: Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and GIS*, 25, 243–257, 1998.

[4] Cabello, S., Liu, Y., Mantler, A., Snoeyink, J.: Testing homotopy for paths in the plane. *In Proc. Annual Symposium On Computational Geometry*, 160–169, 2002.

[5] Chazelle, B.: An algorithm for segment-dragging and its implementation. *In Algorithmica*, 3, 205–221, 1988.

[6] Daneshpajouh, Sh., Abam, M. A., Deleuran, L., Ghodsi, M.: Computing Homotopic Line Simplification in a Plane (Full version). `http://www.daimi.au.dk/~danesh/publications/chls.pdf`

[7] Estkowski, R., Mitchell, J. S.: Simplifying a polygonal subdivision while keeping it simple. *In Proc. Annual Symposium on Computational Geometry*, 40–49, 2001.

[8] Guibas, L.J., Hershberger, J.E., Mitchell, J.S.B., Snoeyink, J.S.: Approximating polygons and subdivisions with minimum link paths. *International Journal of Computatioanl Geometry and Applications*, 3, 383–415, 1993.

[9] Palazzi, L., Snoeyink, J. Counting and reporting red/blue segment intersections. *In CVGIP: Graph. Models Image Process.*, 56(4), Academic Press, Inc., Orlando, FL, USA, 1994.

Geometric Motion Planning: Finding Intersections

Sándor P. Fekete* Henning Hasemann*† Tom Kamphans*‡ Christiane Schmidt*

Abstract

We investigate a new model for mobile agents: Motion planning with geometric primitives, similar to ruler-and-circle constructions in classical geometry. In this first paper on this subject, we consider finding intersection points between two geometric objects using mobile agents that move on these objects. This amounts to finding the rendezvous point of the two agents.

1 Introduction

Algorithms for planning motions for mobile agents have been studied thoroughly in the past decades, both in computational geometry and in the robotics community. Such an algorithm relies on a set of primitives for motion, sensors, and communication that can be performed directly by the agent. Usually, the set of motion primitives models the agent's basic capabilities, most commonly are the primitives *move* and *turn*. As these basic motions are prone to error, it is worth investigating other models (i.e., set of motion primitives).

In this work, we introduce a new model for mobile agents. We assume that the agents are capable of performing geometric primitives, such as *move towards another agent*, *move on a circle*, *follow the ray that is defined by agent a and agent b*. That is, the agents are able to perform constructions similar to ruler-and-circle constructions in classical geometry (without marking trajectories). This model raises two questions: *What can we achieve using this model?* and *How can we implement such geometric primitives given, for example, a real robot?* In this first work on this subject, we focus on one particular aspect of the first question: determining the intersection point of two agents following a trajectory that is modeled by a simple geometric object.

Related Work

Search and rendezvous problems have been considered in many settings (e.g., [1, 5, 2]). Other models for mobile agents have been presented by defining minimal capabilities and investigating solvable tasks and hierarchy relations among different models [8, 4, 7]. Although our search space is a torus in some cases, we do not consider rendezvous on a torus as in [6].

Model

In general, we are given a swarm of non–point-shaped agents. Each agent has perfect communication capabilities, but only limited vision and restricted motion abilities. Motion is limited to a set of primitives that resemble moving on geometric objects.

In our case, we consider two agents with the motion primitives: *Move distance d on a circle* and *Move distance d on a given curve*.

2 Finding Intersections

We are interested in tasks that can be accomplished using our model. A useful building block for modeling complex tasks is to determine the intersection point between two geometric object. This amounts to finding rendezvous strategies for the agents moving on these objects.

Let C_1 and C_2 be two curves in the plane of lengths ℓ_1 and ℓ_2, respectively; A_1 and A_2 be the agents following C_1 and C_2, respectively. Note that we also allow open curves of unbounded length. As the agents are not point shaped, we can safely assume that the agent's minimum travel distance is its diameter (with one exception: In the case of closed curves when the agent returns to its start position after driving around the whole curve, we allow a move with smaller step width). Thus, we have a discrete search space. Transforming the possible positions of the agents into the plane (projecting the positions of A_1 along the X-axis, those of A_2 along the Y-axis) yields an integer grid. The initial position of the agents is defined as $S := (0, 0)$. The topology of the grid depends on whether the curves are closed or not: Two open curves define a disk, one open and one closed curve define a cylinder, and two closed curves define a torus. Now, finding the intersection point of C_1 and C_2 amounts to finding the (first) point $T := (x, y)$ on this grid where the agents meet.

Baeza-Yates et al. [3] considered searching on an infinite integer grid. They showed that any online strategy for finding a point within distance at most k (in L_1-metric) requires at least $2k^2 + O(k)$ steps and presented some strategies that achieve this bound. We use the strategy NSESWSNWN, see Figure 1(ii), for

*Algorithms Group, Braunschweig Institute of Technology, Germany. http://www.ibr.cs.tu-bs.de/alg

†Supp. by 7th Framework Progr. contr. 258885 (SPITFIRE)

‡Supp. by 7th Framework Progr. contr. 215270 (FRONTS)

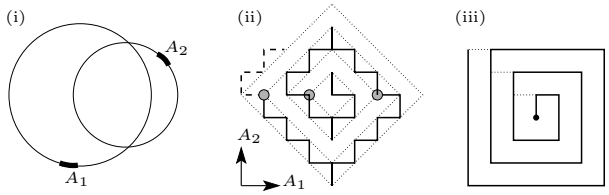


Figure 1: (i) Rendezvous space, (ii)/(iii) Search space. Strategies for (ii) L_1 (NSESWSNWN), (iii) L_∞.

our search. The strategy proceeds by successively visiting all points within distance k, which lie on a *diamond* around the origin; it requires $2k^2+5k+2$ steps. Note that after visiting every point within distance $k-1$, the strategy needs just $4k+3$ additional steps to visit every point within distance k. Although there is a slightly better strategy, we prefer NSESWSNWN because it produces a symmetric search path.

2.1 One-Dimensional Agents

In this section, we consider agents that move on a given curve. For now, the agents are one dimensional; that is, they extend only along the curve. Imagine a snake-like robot following a trail. Further, the agents do *not* move simultaneously.

2.1.1 Closed Curves of Equal Length

Theorem 1 *Let C_1 and C_2 be two curves of length ℓ. Any algorithm that finds the intersection at distance at most k needs at least*
(i) $2k^2+2k-4$ steps (for $k<n=\lceil \ell/2 \rceil$),
(ii) $2n^2+4zn+2n-2z^2-2z-4$ steps if $n<k, k=n+z$.

Proof. Any algorithm must visit every point at distance $\leq k$. Visiting m points takes $m-1$ steps. For $k=0$ we have one point. Each diamond of size k has $4k$ points; thus, we have $1+\sum_{i=1}^{k} 4i = 2k^2+2k+1$ points for $k \leq n$. Beyond n, the number of points per diamond shrinks. Let $z=k-n$, then we have $4(n-z)$ points per diamond, which yields a total of $2n^2+2n+1+\sum_{i=1}^{z} 4(n-i) = 2n^2+4zn+2n-2z^2-2z+1$ points. □

Theorem 2 *Let C_1 and C_2 be two intersecting, closed curves of length ℓ and let A_1 and A_2 be two mobile agents moving on C_1 and C_2, respectively. Let k be the distance to the (closest) rendezvous point, T, of A_1 and A_2. For finding T, the agents need at most*
(i) $2k^2+5k+2$ steps if $k \leq n$,
(ii) $2n^2+4zn+7n-2z^2-3z+2$ steps if $n<k, k=n+z$.

Proof. We use the following rendezvous strategy: We use the NSESWSNWN strategy to explore the search space, until we meet a point that was already explored before. This happens when two tips of the diamond touch each other, because the search space is a torus.

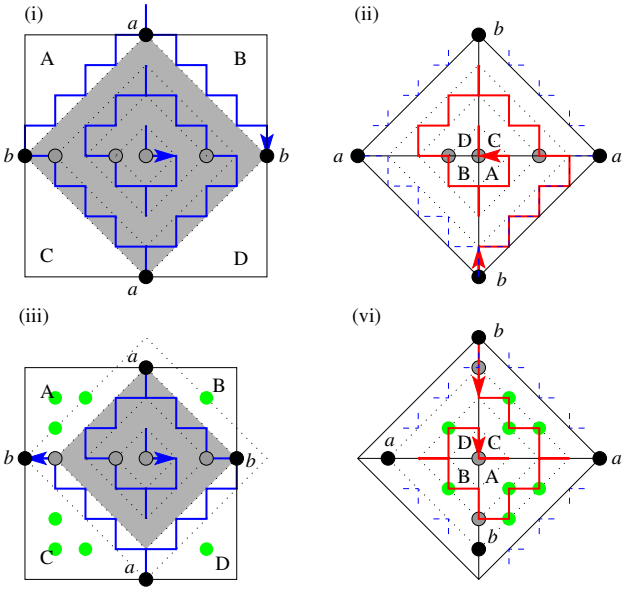


Figure 2: (i) and (iii) searching a diamond with NSESWSNWN until b is visited twice. (ii) and (iv) exploring the remaining diamond. (i) and (ii) show the case for even n, (iii) and (iv) for odd n. The gray dots show the end of one round.

Now we searched a diamond-shaped area and are left with four unsearched triangles, see Figure 2(i). As the search space is a torus, these triangles are connected and form, in turn, another diamond if seen from a different viewpoint, see Figure 2(ii). Thus, we search this diamond with a similar strategy, but starting with the outer layer and proceeding towards the origin of the diamond. Let n be the round when we switch the strategy and k be the current round (that is, we want to explore every point within distance k). When switching the strategies, we need $2n$ additional steps to move from b to the start of the next round if n is even, one step if n is odd. Let $S(k)$ be the total number of steps after finishing round k.
For $k \leq n$ we have $S(k) = 2k^2+5k+2$.
For $n<k<2n$, $k=n+z$, we have $4(n-1-z)+3$ additional steps per layer. Altogether, we have
$S(k) \leq 2k^2+5k+2+2n+\sum_{i=1}^{z} 4\,(n-i-1)+3$
$= 2n^2+4zn+7n-2z^2-3z+2$
□

2.1.2 Closed Curves of Different Length

If the curves have different sizes, the search space is no longer a square projected onto a torus. This means that the expanding diamonds begin to overlap in one dimension, while the other dimension is not yet explored in its total width, see Figure 3.

Let point a be the point where the expanding diamonds touch and let n be the layer in which we reach a. Let m be the layer where b is met.

Theorem 3 *Let C_1 and C_2 be two curves of length ℓ_1 and ℓ_2 ($\ell_1 < \ell_2$), respectively, $n = \lceil \ell_1/2 \rceil, m = \lceil \ell_2/2 \rceil$. Any algorithm that finds the intersection at distance at most k needs at least*
(i) $2k^2 + 2k - 4$ steps if $k \leq n$,
(ii) $2n^2+2n+z'(4n+2)-4$ steps if $n < k = n+z' \leq m$,
(iii) $4mn-2n^2+4nz-2z^2-2z+2m-4$ steps if $k = m+z$.

Proof. Part (i) is the same as in Theo. 1. Beyond the n-diamond, we have $z' = k - n$ $\langle\,\rangle$-shaped layers with $4n + 2$ points each. For (iii), we have the points from Case (ii) with $z' = (m-n)$ and add $\sum_{i=1}^{z} 4(n-i)$. □

Theorem 4 *Let C_1 and C_2 be two intersecting, closed curves of length ℓ_1 and ℓ_2, respectively. Let A_1 and A_2 be two mobile agents moving on C_1 and C_2, respectively. Let k be the distance to the (closest) rendezvous point, T, of A_1 and A_2. For finding T, the agents need at most*
(i) $2k^2 + 5k + 2$ steps if $k \leq n$,
(ii) $6n^2 + 7n + 2^j(n + 3) + 4nz' + 2j - 2$ steps if $n < k = n + z', 2^{j-1} < z' \leq 2^j$,
(iii) $5mn + n^2 + 4nz - 2z^2 - 2z + 4n + 3m + +2\log(m - n) - 2$ if $k = m + z$.

Proof. As above, we start with the NSESWSNWN strategy, until we complete layer n and are located at point b'. Now we switch the strategy. First, we return to a. Then we alternatingly search the areas left and right to the explored diamond, using a stairway-like search pattern, as shown in Figure 3. To keep the relocation costs low, we use the *doubling paradigm*; that is, we double the exploration depth each time we switch sides. We continue until we reach point b, from where we use the inverse NSESWSNWN strategy, as described in the preceding section.

To visit every point within distance k, $n < k < m$, $k = n+z$, we choose j such that $2^{j-1} < z \leq 2^j$. Up to b', we need $2k^2+5k+2$ steps as above. The move from b' to a costs $2n$. Then we explore up to a depth of 2^j using doubling. For doubling step j, we move from a to the point at which we ended after doubling step $j-2$ (plus one step to reach the position for the new stairway), incurring a cost of $2^{j-2}+1$. Then we move on the stairways. Note that each stairway covers two layers; thus, we explore $\frac{1}{2}(2^j - 2^{j-2})$ stairways with $4n$ steps each. Afterwards, we use 2^j+1 steps to return to a. Note that we do not need additional steps between two stairways. Thus, we have for $n < k \leq m$:
$S(k) = 2n^2 + 5n + 2 + 2n + 4n + 4$ for $j = 1$
$S(k) = 2n^2 + 5n + 2 + 2n + 12n + 10$ for $j = 2$
and for $j > 2$:
$S(k) \leq 2n^2 + 5n + 2 + 2n$ (from (i) and move to a)
$+4+6+\sum_{i=3}^{j}(2^i+2^{i-2}+2)+2^{j-1}+1$ (relocation costs)
$+4n\left(1 + 2 + \sum_{i=3}^{j}(2^{i-1} - 2^{i-3})\right) + 4n(k-2^{j-1})$ (stairways)
$= 6n^2 + 7n + 2^j(n + 3) + 4nz' + 2j - 2$

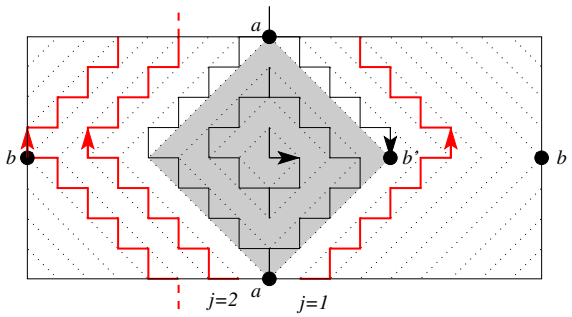


Figure 3: NSESWSNWN and stairways.

After visiting the stairways with the doubling technique, we use an inverse NSESWSNWN strategy as in Case (ii) of Theorem 2. For convenience, assume $m = n+2^j$ and let $k = m+z$. We add $\sum_{i=1}^{z} 4(n-i)-1$ to the result of the preceding case and get $S(k) \leq -2z^2+4nz-4z+6n^2+7n+2^j(n+3)+4nz'+2j-2$. With $2^j = m - n$ we get $S(k) \leq 5mn + n^2 + 4nz - 2z^2 - 2z + 4n + 3m + +2\log(m - n) - 2$. □

2.2 One-Dimensional Agents Moving Simultaneously

In the preceding sections, we assumed that the agents move alternatingly. How do the agents proceed, if they are allowed to move simultaneously? The search space remains the same. If the agents move alternatingly, all points of equal distance to the start lie on a diamond (i.e., a scaled copy of the L_1-unit circle). In this case, points of equal distance form a square (i.e., a scaled copy of the L_∞-unit circle). We consider the case of two curves of equal length.

Theorem 5 *Even if the agents are allowed to move simultaneously, there is an optimal strategy in which the agents move alternatingly.*

Proof. We show that an optimal strategy moves on a rectangular spiral-like search pattern, as shown in Figure 1(iii). Let the target be located at some unknown finite distance, k, from the start point. If an upper bound k' is known to the agent, visiting points in distance $k'+1$ (before every point of distance $\leq k'$ is visited) does not make sense to the agent, because these visits prolong the search path to the unvisited points within distance k'. Thus, if the agent knows no such upper bound, it has to cover each layer of points of same distance i completely before it proceeds to the next layer with points of distance $i+1$. As connecting two successive layers costs one step, the squared spiral is optimal. An axis-parallel move in the search space corresponds to a move of a single agent in the rendezvous space. Thus, the agents move only alternatingly, even if they are allowed to move simultaneously. □

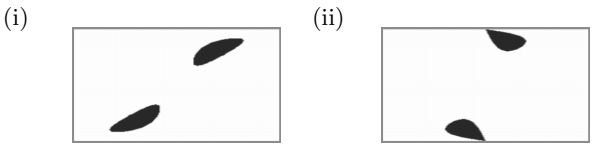


Figure 4: The search space for $R = 1$ and $r = 4$ (i) $d = 3$ (ii) $d = 6$, corner case $d \geq 2r - 2R$, i.e. the rendezvous area is connected.

2.3 Two-Dimensional Agents

Consider two mobile agents shaped as disks of radius R, each moving on one of two intersecting circles of radius r, $r \geq R$, whose centers have distance $d \leq 2r$. We identify the position of the agent on the first circle with the angle α between the circles center and the line between the two circles' centers. Analogously, we express the position of the other agent as an angle β and call the circles of the agents C_α and C_β, respectively. Again, the rendezvous search space is a torus.

Note that, in contrast to the preceding section, modeling the search space as a grid is not sufficient anymore, because now there is an infinite number of rendezvous points, see Figure 4. We observe, however, that the set of rendezvous points for this scenario consists of no more than two simply connected components, one for each intersection point. For $d \geq 2r-2R$ or $d \leq 2R$, we have only one simply connected component. In the rest of this section, we will assume that none of these special cases occurs.

Searching for a point on a torus is quite involved; thus, we want to find a convex region of a certain size. This allows us either to inspect a finite set of points on a grid or to move on an Archimedean spiral.

Lemma 6 *In the search space, there is a square of size at least $2R \times 2R$ such that all points inside the square are rendezvous points.*

Proof. First, observe the line, M, through the centers of C_α and C_β. For reasons of symmetry, in the following we consider only one side of M. Construct the line segment, L, of length $2R$ that is parallel to M and whose end points lie on C_α and C_β, see Figure 5. Let p_α and p_β be the endpoints of L lying on C_α and C_β, respectively. Let q_α and q_β be the intersection points of C_α and C_β and the lines perpendicular to L through p_β and p_α, respectively. Note that $|\overline{p_\alpha q_\beta}| = |\overline{q_\alpha p_\beta}|$. We consider two cases:

Case 1: $|\overline{p_\alpha q_\beta}| \leq 2R$. All pairs of points with α on the arc from p_α to q_α and β on the arc from p_β to q_β define a rendezvous point.

Case 2: $|\overline{p_\alpha q_\beta}| > 2R$. We parallely move both $\overline{p_\alpha q_\beta}$ and $\overline{q_\alpha p_\alpha}$ towards the intersection point, such that $|\overline{p'_\alpha q'_\beta}| = |\overline{q'_\alpha p'_\beta}| = 2R$ holds. Now, all pairs of points with α on the arc from p'_α to q'_α and β on the arc from p'_β to q'_β define a rendezvous point.

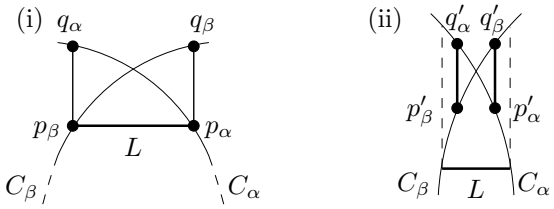


Figure 5: Construction of $p/q_{\alpha/\beta}$. The bold lines have length $2R$. (i) Case 1, (ii) Case 2.

In both cases, the arcs for α and β have length greater than or equal to $2R$. Thus, there is a $2R \times 2R$ square of rendezvous points in the search space. □

2.4 Other Topologies

If one of the curves is open/infinite, the search space is no longer a torus. In these cases, we search the rendezvous point with strategies combining spiral searches and doubling. In the case of finite, open curves it may happen that one end of the search space is fully explored, while other parts are unexplored. In this case, we use a meandering search path. We leave the details to the full version of this paper.

Acknowledgements

We thank Friedhelm Meyer auf der Heide, Bastian Degener, and Barbara Kempkes for helpful discussions.

References

[1] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous*. Kluwer Academic Publications, 2003.

[2] E. J. Anderson and S. P. Fekete. Two-dimensional rendezvous search. *Oper. Res.*, 49:107–118, 2001.

[3] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Inform. Comput.*, 106:234–252, 1993.

[4] J. Brunner, M. Mihalak, S. Suri, E. Vicari, and P. Widmayer. Simple robots in polygonal environments: A hierarchy. In *Proc. Algosensors*, 2008.

[5] R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen. Competitive online approximation of the optimal search ratio. *Siam J. Comput.*, pages 881–898, 2008.

[6] E. Kranakis, D. Krizanc, and E. Markou. Mobile agent rendezvous in a synchronous torus. In *LATIN*, pages 653–664, 2006.

[7] J. M. O'Kane and S. M. LaValle. Dominance and equivalence for sensor-based agents. In *Proc. 22nd National Conf. Artif. Intell.*, pages 1655–1658, 2007.

[8] S. Suri, E. Vicari, and P. Widmayer. Simple robots with minimal sensing: From local visibility to global geometry. *Int. J. Robot. Res.*, 27:1055–1067, 2008.

Kinetic Red-blue Minimum Separating Circle

Yam Ki Cheung, Ovidiu Daescu, and Marko Zivanic
Department of Computer Science
The University of Texas at Dallas
Richardson, TX USA
Email: {ykcheung,daescu,mxz052000}@utdallas.edu

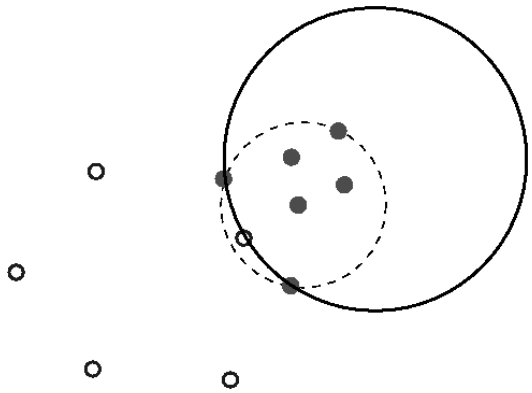


Figure 1: Minimum red enclosing circle (dashed) and red-blue separating circle (solid).

1 Introduction

Let $\mathcal{R}$ and $\mathcal{B}$ be two finite sets of points in $\mathbb{R}^2$, of size $|\mathcal{R}| = n$ and $|\mathcal{B}| = m$, respectively. We refer to $\mathcal{R}$ as the set of red points and to $\mathcal{B}$ as the set of blue points. In [4] the authors define a constrained version of the circular separability problem, called the *minimum separating circle problem*, as follows: Let $\mathcal{S}$ denote the set of circles such that each circle in $\mathcal{S}$ encloses all points in $\mathcal{R}$ while having the smallest number of points of $\mathcal{B}$ in its interior. The goal is to find the smallest circle in $\mathcal{S}$, called the *minimum separating circle* and denoted by $C_{\mathcal{B}}(\mathcal{R})$. See Figure 1 for an illustration.

The problem has applications in military planning. It can be used to determine the best location to deploy an explosive and the amount needed so that all enemy forces, represented by red points, will be impacted while minimizing civilian casulties (blue points). It is also applicable in determining the best set-up of communication devices such that all red devices stay connected and as few blue devices as possible can intercept their communication. Two algorithms for the static version of this problem have been proposed by Bitner et al. [4].

In practice, however, it is possible that not all points (targets) are stationary. In this paper, we study a kinetic version of the red-blue minimum separating circle problem, in which all points are stationary except one red point, which moves along a linear trajectory with constant velocity. We want to find the locus of the minimum separating circle over a period of time.

For the case when the two point sets can be separated, Fisk [6] gave a quadratic time and space algorithm to compute the minimum separating circle. The result was later improved to optimal linear time and space by O'Rourke et. al. [8].

To the best of our knowledge the kinetic version of the minimum separating circle problem has not been studied in the past, but there is a significant number of publications on related topics. Atallah [1] introduced the concept of kinetic computational geometry in a seminal paper on this topic. Basch et al. [3] introduced a set of kinetic data structures that can be used to maintain the convex hull of a moving set of points. Ross [10] gived an algorithm for maintaining the nearest-point Voronoi diagram of a kinetic data set. He presented an update algorithm for the topological structure of the Voronoi diagram of moving points, using $O(\log n)$ time for each change. Demaine et al. [5] presented a kinetic data structure that calculates the minimum spanning circle for a moving set of points. Banik et al. [2] solved the minimum enclosing circle problem of a fixed set of points and one moving point. Their algorithm computes the locus of the center of the minimum enclosing circle in linear time. Rahmati et al. [9] presented a kinetic data structure for the maintenance of the minimum spanning tree on a set of moving points in $\mathbb{R}^2$.

2 Preliminaries

We start by briefly discussing an algorithm proposed by Bitner et al. [4] for the static version of the minimum separating circle problem. The algorithm is based on a sweep procedure on the edges of the farthest neighbor Voronoi diagram $FVD(\mathcal{R})$ of $\mathcal{R}$.

Lemma 1 *[4] The smallest separating circle must pass through at least two points from $\mathcal{R}$.*

It follows that the minimum separating circle is either the smallest enclosing circle of $\mathcal{R}$, which can be found in linear time [7], or a circle which passes

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

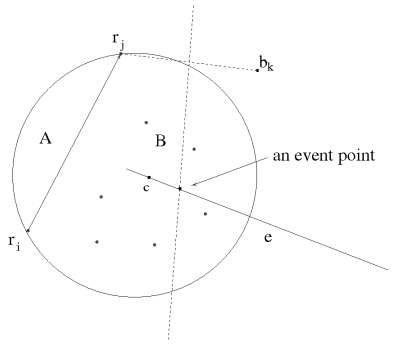


Figure 2: An event point on edge e_{ij}.

through two points from $\mathcal{R}$ and one point from $\mathcal{B}$. Thus, the center of a minimum separating circle lies on an edge of $FVD(\mathcal{R})$.

Following Lemma 1, Bitner et al. [4] proposed a sweep algorithm for computing the minimum separating circle.

Consider a Voronoi edge e_{ij}, defined by two red points r_i and r_j. The sweep procedure on e_{ij} is initialized by constructing an enclosing circle C of $\mathcal{R}$ which passes through r_i and r_j and has the smallest possible radius. This happens at one endpoint of e_{ij}. Let $c \in e_{ij}$ be the center of C. C is grown by sweeping c along e_{ij} and keeping r_i and r_j on the circumference of C.

A point $E \in e_{ij}$ is an event point if, when c sweeps through E, the circle C sweeps through a blue point (see Figure 2 for an illustration). The number of blue points enclosed by C is updated at each event point. The sweep procedure on e_{ij} terminates when the other endpoint of e_{ij} is reached. To total number of event points over $FVD(\mathcal{R})$ is $O(mn)$.

Lemma 2 *[4] A point from $\mathcal{B}$ defines at most one exit event point on $FVD(\mathcal{R})$, and such exit event point can be found in $O(\log n)$ time.*

3 The minimum separating circle with one mobile red point

In this section, we study the minimum separating circle problem for two point sets $\mathcal{R}$ and $\mathcal{B}$, such that all points are stationary except one red point p, which is moving along a linear trajectory with constant velocity. We show how to find the locus of the center of the minimum separating circle $C_{\mathcal{R}}(\mathcal{B})$ over a period of time.

Following Lemma 2, at any time instant t, we can find the minimum separating circle by computing the $FVD(\mathcal{R})$ and checking $O(m)$ exit event points. To find the locus of the minimum separating circle over a period of time, it is sufficient to keep track of the (trajectory of) the exit event associated with each blue point as well as the number of blue points enclosed by the corresponding candidate separating circle. Since one red point is mobile, the topology of $FVD(\mathcal{R})$ changes continuous. In order to keep track of the trajectory of each exit event point, we define the following four classes of events: 1) the appearance/disappearance of a vertex on $FVD(\mathcal{R})$; 2) the appearance/disappearance of a vertex on the boundary of the convex hull $CH(\mathcal{R})$ of $\mathcal{R}$; 3) the exit event point associated with a blue point moves to another Voronoi edge, and 4) the candidate separating circle associated with an exit event point encloses/excludes a blue point.

To avoid ambiguity, we refer to these four classes of events as *instant events*, distinguishing from the event points introduced in Section 2. Notice that case 1 and case 2 instant events indicate that the topology of the $FVD(\mathcal{R})$ need to be updated. Case 3 instant events indicate that the trajectory of some exit event point changes, i.e. it moves along a different line. And case 4 indicates that the number of blue points enclosed by some candidate separating circle needs to be updated.

Lemma 3 *The locus of the center of the minimum separating circle can be discontinuous.*

Lemma 4 *The locus of the center of the minimum separating circle consists of a set of halflines, line segments, or points.*

3.1 Case 1 instant events

These are time instants when a Voronoi edge appears or disappears from $FVD(R)$.

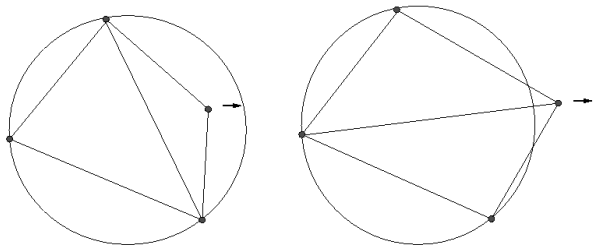


Figure 3: Update of Delaunay triangulation of four points by edge swapping.

Lemma 5 *There are at most $O(n)$ instant events in case 1. For each instant event, $FVD(\mathcal{R})$ can be updated in constant time.*

Proof. Instead of working on $FVD(\mathcal{R})$ directly, we turn our attention to the dual graph of $FVD(\mathcal{R})$, the Delaunay triangulation $DT(\mathcal{R})$.

The topology of $DT(\mathcal{R})$ could change when four red points, including the mobile red point p, on the boundary of $CH(R)$ are cocircular, or more specifically, when p enters/leaves a circle passing through three fixed red points, and enclosing all fixed red

points in $\mathcal{R}$. See Fig. 3. Observe that the center of each such enclosing circle defines a vertex of $FVD(\mathcal{R}/\ p)$, so there are $O(n)$ such enclosing circles. $DT(\mathcal{R})$ can be updated in constant time by edge swapping. □

3.2 Case 2 instant events

These are time instants when a vertex of $CH(R)$ appears or disappears.

Lemma 6 *There are $O(n)$ instant events in case 2, and each event can be identified in constant time.*

Proof. Case 2 instant events occur when p passes through the intersection between a line supporting some edge of $CH(\mathcal{R}/\ p)$ and the trajectory of p. Hence, each instant event can be identified in constant time. See Fig. 4 for an illustration

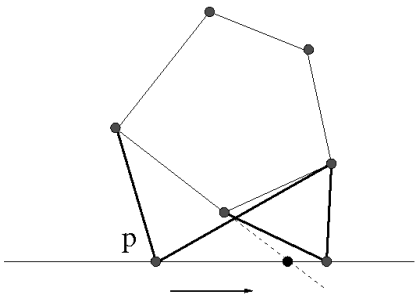


Figure 4: A vertex of $CH(R)$ appears.

□

3.3 Case 3 instant events

These are time instants when an exit event point moves to another edge of $FVD(\mathcal{R})$.

Lemma 7 *Each case 3 instant event can be identified in constant time.*

Lemma 8 *There are $O(nm)$ instant events in case 3.*

3.4 Case 4 instant events

These are time instants when the candidate separating circle centering at an exit event point encloses/excludes a blue point.

Lemma 9 *Each case 4 instant event can be found in linear time.*

Proof. Let e be a Voronoi edge defined by a fixed red point r_i and the mobile red point p and assume the exit event point of a blue point b lies on e. The candidate separating circle centering at the exit event point of b is the circumcircle of r_i, p, and b. The locus of the exit event point is a line segment supported by the bisector between r_i and b. We can keep track of the number of blue points enclosed by the candidate separating circle by running a sweep algorithm along the locus of the exit event point. This sweep algorithm resembles the one used to compute the minimum separating circle of two sets of fixed points. The only difference is that the center of the candidate separating circle moves along the bisector between r_i and b instead of a Voronoi edge.

The case 4 instant event is the moment when the exit event point crosses an event point defined on the bisector between r_i and b. □

Lemma 10 *There are at most $O(m^2n)$ case 4 instant events.*

Proof. As shown in Lemma 8, one exit event point can traverse $O(n)$ Voronoi edges. For each edge traversed by the exit event point there can be $O(m)$ event points on the bisector between the blue point and the fixed red point which defines the Voronoi edge. Each event point corresponds to one case 4 instant event. Hence, each exit event generates at most $O(mn)$ case 4 instant events. □

3.5 Finding the locus of the center of the minimum separating circle

First, we need to determine the exact trajectory of each exit event point. If in a given time interval $[t_1, t_2]$, the exit event point of a blue point b lies on a Voronoi edge defined by two fixed red points, the exit event point is stationary. However, if the Voronoi edge is defined by one fixed red point a, and the mobile red point p, then the corresponding exit event point is the intersection between the bisector $B(a, b)$ between a and b and the bisector $B(b, p)$ between b and p. The trajectory of p can be expressed parametrically as $x = v_x t$ and $y = m_p v_x t + q_p$, where v_x is a constant and m_p and q_p are the slope and the intercept of the trajectory of p, respectively. It is not difficult to show that the trajectory of the intersection between $B(a, b)$ and $B(b, p)$ can be expressed as:

$$x = \frac{t^2 + c_1 t + c + 2}{c_3 t + c_4},$$

$$y = \frac{t^2 + c_1 t + c + 2}{c_3 t + c_4} m' + q',$$

where c_1, c_2, c_3, c_4 are constants and m' and q' are the slope and intercept of $B(a, b)$, respectively.

Once the trajectory of the exit event point is known, the function of the square radius of the corresponding candidate separating circle is a function $f_b(t)$ of constant degree.

Following Lemma 8, each exit event point cannot cross more than $O(n)$ Voronoi edges. For a time interval $[t_{init}, t_{end}]$, $f_b(t)$ consists of $O(n)$ pieces of curves

or horizontal line segments if the exit event point is stationary.

Plotting all functions $f_b(t)$ for $t \in [t_{init}, t_{end}]$ and $b \in \mathcal{B}$ on the same coordinate system gives us an arrangement H of curves of complexity $O(nm^2)$, since all functions are x-monotone and two such functions intersect no more than $O(n)$ times. H gives us the relative size between all candidate circles over time.

However, we also need to consider the blue points enclosed by the candidate circles. We further decompose H by dividing each function $f_b(t)$ at every case 4 instant event t_o generated by b by introducing a vertex at $(t_o, f_b(t_o))$. As a result, each portion of $f_b(t)$ on the new arrangement H' represents the square radius of the candidate circle for a time interval during which the blue points enclosed by the circle remain the same. The new arrangement H' has complexity $O(m^2n)$. We call each portion of $f_b(t)$ on H' a *simple curve*.

Let the blue point count of a simple curve on H' be the number of blue points enclosed by the corresponding candidate circle. The last step to compute the locus of the minimum separating circle is to extract the lower envelope of curves with the lowest blue point count. Each curve on the lower envelope gives us the minimum separating circle for the interval spanned by the curve, hence, the locus of the center of the minimum separating circle.

We use a plane sweep to extract such lower envelope. We sweep H' by a vertical line. At any moment, the sweep line intersects with at most m functions $f_b(t)$, for $b \in \mathcal{B}$. Each function is indexed by its blue point count at the current moment. We build a hash table for functions intersected by the sweep line. For each entry of the hash table, all functions are maintained in a balanced tree by the order intersected by the sweep line. Note that we only have to update the hash table at vertices of H'. If two functions with the same blue point count intersect, we need to exchange their position in the corresponding tree. If the sweep line crosses a vertex introduced by a case 4 instant event, the blue point count of a function changes. The corresponding function will be moved to the appropriate entry of the hash table. It takes $O(\log m)$ time to update the hash table for each instance.

Theorem 11 *The locus of the center of the minimum separating circle has complexity of $O(m^2n)$ and can be found in $O(m^2n \log m)$ time.*

References

[1] Mikhail J. Atallah. Dynamic computational geometry (preliminary version). In *FOCS*, pages 92–99, 1983.

[2] Aritra Banik, Bhaswar B. Bhattacharya, and Sandip Das. Minimum enclosing circle of a set of fixed points and a mobile point. In *Proceedings of the Workshop on Algorithms and Computation*, New Delhi,India, 2011.

[3] Julien Basch, Leonidas J. Guibas, Craig Silverstein, and Li Zhang. A practical evaluation of kinetic data structures. In *Symposium on Computational Geometry*, pages 388–390, 1997.

[4] Steven Bitner, Yam Ki Cheung, and Ovidiu Daescu. Minimum separating circle for bichromatic points in the plane. In *ISVD*, pages 50–55, 2010.

[5] Erik Demain, Sarah Einsenstat, Leonidas Guibas, and Andre Schulz. Kinetic minimum spanning circle. In *Proceedings of the Fall Workshop on Computational Geometry*, New York, NY, USA, 2010.

[6] S. Fisk. Separating point sets by circles, and the recognition of digital disks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 554–556, July 1986.

[7] N. Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

[8] J. O'Rourke, S. Kosaraju, and N. Megiddo. Computing circular separability. *Discrete Computational Geometry*, 1:105–113, 1986.

[9] Zahed Rahmati and Alireza Zarei. Combinatorial changes of euclidean minimum spanning tree of moving points in the plane. In *CCCG*, pages 43–45, 2010.

[10] Thomas Roos. Voronoi diagrams over dynamic scenes. *Discrete Applied Mathematics*, 43(3):243–259, 1993.

Persistent Homology Computation with a Twist

Chao Chen* Michael Kerber†

Abstract

The persistence diagram of a filtered simplicial complex is usually computed by reducing the boundary matrix of the complex. We introduce a simple optimization technique: by processing the simplices of the complex in decreasing dimension, we can "kill" columns (i.e., set them to zero) without reducing them. This technique completely avoids reduction on roughly half of the columns. We demonstrate that this idea significantly improves the running time of the reduction algorithm in practice. We also give an output-sensitive complexity analysis for the new algorithm which yields to sub-cubic asymptotic bounds under certain assumptions.

1 Introduction

Persistent homology is a quickly-growing area of research in the analysis of topological spaces. Substantial progress, both theoretical and practical, has been made during the last decade; we refer to [3] for a recent textbook on the topic.

The classical way of computing the persistence of a simplicial complex (or more precisely, a filtration of it) is by reduction [4]: the boundary matrix of the complex is transformed by column operations until each column either turns zero, or has a unique lowest nonzero entry. For a complex with n simplices, the algorithm runs in cubic time in the worst case; an example where this bound is actually achieved has been presented in [6]. However, it has been observed that the algorithm rather behaves linear in practical applications [7, 1].

We present an optimization of the persistence algorithm. The idea is that the reduction of a column that corresponds to a negative simplex (one that destroys an homology class) also reveals the simplex that creates the corresponding class. Since that column is known to be zero after reduction, we can simply set it to zero ("kill it") without applying any column operation on it. We change the order of column reduction in the algorithm to profit from this trick as much as possible.

Our algorithm permits an output-sensitive complexity analysis. Writing d for the dimension of the complex, P for the persistence pairs of the filtration (that is, the indices of creator/destroyer pairs), and E for the set of homology classes of the underlying simplicial complex, we obtain a bound of

$$O\left(d \log n (\#E \cdot \sum_{(i,j)\in P} (j-i) + \sum_{(i,j)\in P} (j-i)^2)\right).$$

Although our variant does not improve the worst case bound (and even worsens it by a factor of $d \log n$), the bound still shows that the algorithm can perform sub-cubic, and even sub-quadratic if the total index persistence, namely, the squared sum of indices differences of persistence pairs, is small.

We further investigate the case of cubical data, which is the common format in computer vision and visualization: the space is a cubical subset of Euclidean space tiled into unit cubes; the filtration is done with respect to a function f that assigns a value to the center of each unit cube. Based on our complexity bound above, we can prove a running time of $O(c^2 n \log n)$ for computing persistence on such data, where c is the number of critical points of f. We also compare the practical performance of our implementation with the classical approach as well as with the sophisticated cohomology algorithm of the Dionysus library and observe a better running time of our optimization for cubical data in $\mathbb{R}^3$.

2 Background

Let $K = \{\sigma_1, \ldots, \sigma_n\}$ denote a simplicial complex of dimension d. We assume an ordering on the simplices such that for each $i \leq n$, $K_i := \{\sigma_1, \ldots, \sigma_i\}$ is a simplicial complex again. The chain $\emptyset = K_0 \subset K_i \ldots \subset K_n = K$ is called a *filtration* of K. Normally, such a filtration is defined according to a function $f : K \to \mathbb{R}$ that orders the simplices of K by function value.

For $0 \leq i \leq n$ and $0 \leq p \leq d$, we denote the p-th homology group of K_i by $\mathsf{H}_p(K_i)$, and we write $\mathsf{H}(K_i)$ for the direct sum of all homology groups of K_i. The rank of $\mathsf{H}_p(K_i)$ is called the p-th Betti number $\beta_p(K_i)$. We obtain K_i by adding the simplex σ_i into K_{i-1}. Denoting d_i as the dimension of σ_i, there are two possible changes of the homology due to the addition of σ_i, either a class of dimension d_i is created or a class of dimension $d_i - 1$ is destroyed. In the former case, we call σ_i a *positive* simplex, or a *creator*; in the latter case, it is called *negative*, or a *destroyer*.

To each negative simplex σ_j, we can associate a unique positive simplex σ_i with $i < j$ such that σ_j

*IST Austria, Klosterneuburg, Austria; Vienna University of Technology, Vienna, Austria, chao.chen@ist.ac.at

†IST Austria, Klosterneuburg, Austria, mkerber@ist.ac.at

destroys the homology class that was created when σ_i was added in the filtration (see [3] for a precise definition). We call (σ_i, σ_j) a *persistence pair* and $j-i$ its *index persistence*. Note that there are simplices that are not paired, namely those (positive) simplices that create homology classes of the simplicial complex K; we call them *essential* simplices. By definition, every simplex of K either belongs to a persistence pair or is essential.

Reduction Algorithm. The boundary of a simplex σ in dimension d_σ is the set of its faces in dimension $d_\sigma - 1$. For $d_\sigma = 0$, the boundary is empty, otherwise, it consists of exactly $d_\sigma + 1$ simplices. The boundary matrix $\partial \in (\mathbb{Z}_2)^{n\times n}$ of a filtered complex K is a $n \times n$ matrix where the j-th column represents the boundary of σ_j, that is $\partial_{i,j} = 1$ if and only if σ_i belongs to the boundary of σ_j. In this case, σ_i must belong to the complex before σ_j is added, so ∂ is an upper-triangular matrix.

For $0 \neq M_j = (m_1, \ldots, m_n) \in \mathbb{Z}_2^n$, we set $\mathrm{low}(M_j) := \max\{i = 1, \ldots, n \mid m_i = 1\}$. For $M_j = 0$, $\mathrm{low}(M_j)$ is undefined. A column operation of the form $M_j \leftarrow M_j + M_k$ is called *reducing* if $k < j$ and $\mathrm{low}(M_j) = \mathrm{low}(M_k)$. A matrix is called *reduced* if no reducing column operation can be performed on it. We call a matrix R a *reduction* of M if R is reduced and arises from a sequence of reducing column operations from M.

A reduction R of the boundary matrix ∂ as above yields the complete information about the persistent homology of the complex. Define

$$\begin{aligned} P &:= \{(i,j) \mid R_j \neq 0 \wedge i = \mathrm{low}(R_j)\} \\ E &:= \{i \mid R_i = 0 \wedge \mathrm{low}(R_j) \neq i \forall j = 1, \ldots, n\}. \end{aligned}$$

Then, $(i,j) \in P$ if and only if (σ_i, σ_j) form a persistence pair of the underlying filtrated complex. Moreover, $i \in E$ if and only if σ_i is essential [3]. In particular, this information does not depend on the choice of the reduction. The simplest way of reducing ∂ is to process from left to right and to reduce a column completely by adding columns from its left (Algorithm 1). A lookup table can be used to identify the next column to be added in constant time. It can be observed immediately that the running time is cubic in n, and cubic running time is indeed necessary for certain input filtrations as presented in [6].

Algorithm 1 Left-to-right persistence computation

```
1: procedure PERSISTENCE_LEFT_RIGHT(∂)
2:     R ← ∂; L ← [0, ..., 0]                    ▷ L ∈ Z^n
3:     for j = 1, ..., n do
4:         while R_j ≠ 0 ∧ L[low(R_j)] ≠ 0 do
5:             R_j ← R_j + R_{L[low(j)]}
6:         if R_j ≠ 0 then L[low(R_j)] ← j
7:     return R
```

3 Reduction by killing

Let $R_j \neq 0$ a column of a reduction of ∂ and $i = \mathrm{low}(R_j)$. Recall that σ_j kills the homology class that is created by σ_i. The key observation exploited in our new algorithm is that in this case, R_i must be zero, since σ_i is in particular a positive simplex. So, we can just set the i-th column of ∂ to zero and have it reduced, without doing any further column operation on it. We say in this case that we *kill* column i.

Reconsidering Algorithm 1, the killing idea does not save any column operations, because whenever a column is identified as positive, it is already reduced due to the left-to-right traversal strategy. Therefore, we change the reduction order of the columns and proceed in decreasing dimensions: setting $d := \dim K$, we first reduce columns that correspond to d-simplices (from left to right), then columns that correspond to $(d-1)$-simplices, and so on. Indeed, if the reduced j-th column has lowest entry i, the corresponding simplex σ_i has dimension $\dim \sigma_j - 1$, so in the moment when ∂_j (namely, the j-th column of ∂) is reduced, ∂_i has not been processed yet. Algorithm 2 summarizes the new method. The dimension of the complex K is passed as a second argument. We also assume that each column stores the dimension of the simplex that it represents in an additional data field. We call this the *simplex-dimension* of a column. By an inductive argument on the simplex dimension, one can prove that Algorithm 2 and Algorithm 1 return the same reduction matrix R.

Algorithm 2 Decreasing dimension persistence computation

```
1: procedure PERSISTENCE_DECR_DIM(∂, d)
2:     R ← ∂; L ← [0, ..., 0]                    ▷ L ∈ (Z)^n
3:     for δ = d, ..., 1 do
4:         for j = 1, ..., n do
5:             if M_j has simplex-dimension δ then
6:                 while R_j ≠ 0 ∧ L[low(R_j)] ≠ 0 do
7:                     R_j ← R_j + R_{L[low(j)]}
8:                 if R_j ≠ 0 then
9:                     i ← low(R_j)
10:                    L[i] ← j
11:                    R[i] ← 0                   ▷ Kill column i
12:    return R
```

4 Analysis

The same analysis as before also shows a worst-case bound of $O(n^3)$ for Algorithm 2. The example from [6] can be adapted to show that cubic running time can also be achieved for this algorithm. However, we will derive a complexity bound that depends on the index persistence of the pairs in the complex, and the number of essential classes of the complex.

We store a matrix $M \in (\mathbb{Z}_2)^{n\times n}$ as an array of

size n, each entry representing a column. A column is stored as a balanced binary search tree storing the indices of nonzero entries as nodes. This way, an operation of the form $M_j \leftarrow M_i + M_j$ can be performed with $O(\#M_i \log(\#M_i + \#M_j)) = O(\#M_i \log n)$ operations, where $\#M_i$ is the number of nonzero entries in column i. Compared to the usual list-representation of columns, the worst case complexity worsens by a logarithmic factor when using trees. However, the complexity of an operation only depends logarithmically on the column to be reduced; this is advantageous when a column accumulates more and more entries by adding small columns during the reduction.

Lemma 1 *Algorithm 2 (as well as Algorithm 1) reduces the j-th column in time $O(\sum_{i=1}^{j-1} \#R_i \log n) = O(\#R \log n)$, where $\#R$ is the total number of nonzero entries in the final reduced matrix R.*

Proof. In the reduction process for a single column, we add only columns from the left to it, and each one at most once. Moreover, all columns which can be added are already reduced. □

The next goal is to bound $\#R$. The crucial observation for that is the following:

Lemma 2 *Let $R_j \neq 0$ be a reduced column and $i = \mathrm{low}(R_j)$. Then, R_j is a linear combination of the columns $\partial_{i+1}, \ldots, \partial_j$. In particular, $\#R_j \leq (d+1)(j-i)$.*

Proof. Assume that the claim is true for any nonzero column with index smaller than j. Initially, R_j is set to ∂_j which is clearly a linear combination. During the reduction, a column R_k is added to R_j only if $k < j$ and also only if $\ell := \mathrm{low}(R_k) > i$, because otherwise $\mathrm{low}(R_j) < i$ at the end. It follows that $i < \ell < k < j$ and so, R_k is by induction a linear combination of $\partial_{\ell+1}, \ldots, \partial_k$. The number of 1's in any column ∂_k is bounded by $d+1$. Thus, only up to $(d+1)(j-i)$ 1's can appear in R_j. □

Corollary 3 $\#R \leq (d+1) \sum_{(i,j)\in P} (j-i)$

We could simply multiply the bound from Lemma 1 by n to obtain an output sensitive bound both for Algorithm 1 and 2. However, we can further improve on this by exploiting that we zero out many columns in Algorithm 2 instead of reducing them. As a first step, we refine the argument from Lemma 2 to derive a more adaptive bound for non-zero columns of R:

Corollary 4 *Let $R_j \neq 0$ be a reduced column and $i = \mathrm{low}(R_j)$. Algorithm 2 (as well as Algorithm 1) reduces ∂_j to R_j in $O(d \log n (j-i)^2)$.*

Proof. Recall from the proof of Lemma 2 that we only add R_k to R_j if $\ell = \mathrm{low}(R_k)$ satisfies $i < \ell < k < j$. So, the number of ones in R_k is bounded by $(d+1)(k-\ell) \leq (d+1)(j-i)$. Since at most $(j-i)$ column operations of that form must be performed for R_j, the bound follows. □

Theorem 5 *Algorithm 2 has a total running time of*

$$O(d \log n (\sum_{(i,j)\in P} (j-i)^2 + \#E \sum_{(i,j)\in P} (j-i))).$$

Proof. Note that the columns of ∂ are in one-to-one correspondence to the simplices of K. It thus makes sense to talk about negative, positive and essential columns of ∂. We divide the columns in three different classes: for negative columns, we can bound the cost by $O(d \log n \sum_{(i,j)\in P} (j-i)^2)$ by Corollary 4. For essential columns, Lemma 1 yields a total cost of $O(\#E \log n \#R) = O(d \log n \#E \sum_{(i,j)\in P} (j-i))$. The remaining columns are positive, but inessential columns. By the killing idea, they are zeroed out before any operation is performed on them, thus, the cost for them is zero. □

5 Cubical Complex

We give an example of how this bound leads to subcubic bounds for special input types. We refer to [3, §VI] for definitions of concepts introduced in this section. Let d be a fixed constant from now and consider a regular cubical grid in d dimensions. We let f denote a function that assign a (different) real value to each grid point. By consistently triangulating each d-cube of the grid without introducing new vertices (e.g., as done in [1]), we can extend f to a piecewise linear function. Depending on its *lower star*, each vertex is either regular or critical; we let c denote the number of critical points of f. We filter the complex according to the *lower star filtration* with respect to f. Due to the cubical structure, the number of simplices in each lower star is bounded by a constant (which is exponential in d).

The final complex is homeomorphic to a d-ball, so there are no essential classes except for the connected component; hence we only have to consider the cost of negative simplices. We distinguish two cases: for persistence pairs that are created and destroyed within the same lower star, the index persistence is bounded by a constant, and since there are up to $n/2$ such pairs, the total cost for them is $O(n \log n)$ with Corollary 4 (with n being the size of the complex). Second, we consider pairs that span over more than one lower star. There are $O(c)$ such pairs, and by Lemma 1 and Corollary 3, we can reduce them in $O(c\#R \log n)$ operations, where $\#R$ is the number of ones in the boundary matrix. Note that pairs that live in only one lower star cause at most a constant number of

ones in R, and pairs that span over more lower stars cause up to n ones in R. This yields a bound of $\#R = O(n + cn)$, and thus a total running time of $O(c^2 n \log n)$ for the reduction algorithm.

Our bound shows that when the complex is subdivided regularly (for instance, by a barycentric subdivision), the running time of computing persistence scales with a $n \log n$ factor which matches general observations in practice. However, our idealistic analysis can only be a first step to investigate this behavior, because introducing noise usually increases the number of critical points in the complex.

6 Experiments

We verify our optimization in practical data. In specific, we focus on cubical data, namely, when the topological space of interest is a subset of Euclidean space (e.g. a hypercube), and the function is sampled uniformly. This class of data is common in image processing and visualization. In specific, we use 3-dimensional data from the *Volvis voxel data repository*[1]. A sample of the results are shown in Table 1.

We triangulate the domain with the *Freudenthal triangulation* [5]. In this case, the lower star of each vertex has a bounded size, which is exponential to the dimension of the domain. Although originally given as $256 \times 256 \times 256$, we downsized the data to $100 \times 100 \times 100$, which leads to simplicial complexes with 25.4 million simplices.

We compare our method (KIL) with two other implementations, the standard reduction algorithm (STA), and the cohomology-reduction algorithm (COH) in the *Dionysus*[2] implementation by Morozov [2]. We compare in terms of both execution time and number of basic add operations. The execution time includes both matrix reduction time and the time for building boundary matrices based on given filtrations. We use `std::vector` instead of a binary search tree to represent each column because the former has shown better practical behavior.

The testing platform of our experiments is a six-core AMD Opteron(tm) processor 2.4GHz with 512KB L2 cache per core, and 66GB of RAM, running Linux. The code runs on a single core.

The experiments show a significant improvement of our method over the standard algorithm and over cohomology reduction. We also notice that the factor of improvement highly depends on the particular input instance. We remark that cohomology reduction is even slower than the standard algorithm in some cubical data. Although cohomology reduction appears to have fewer add operations, it incurs more overhead due to involved data structures in Dionysus.

We conclude by reporting the performance on two non-cubical data by Morozov [2], namely, the alpha complex of uniform samples of a torus embedded in 3-dimensional Euclidean space (0.6 million simplices), and the 4-skeleton of the rips complex of the mumford data (2.4 million simplices). Although our algorithm again improves over the standard persistence algorithm, it is slower than cohomology reduction algorithm on this data. Hence, we observe that the two methods (KIL and COH) have different behaviors over different data; a deeper understanding of this is needed and will be our future work.

Execution time (minute)				
	aneurism	bonsai	foot	skull
KIL	1.02	1.10	1.16	1.45
STA	6.95	7.63	6.54	9.45
COH	1.47	19.76	59.20	95.09
Number of Add Operations (million)				
	aneurism	bonsai	foot	skull
KIL	472.8	104.2	439.9	692.1
STA	24436.8	33664.6	27410.6	66693.3
COH	56.2	3276.1	7064.2	15362.3

Table 1: Performance on cubical data.

	Time(minute)		Add Operations(million)	
	Torus	Mumford	Torus	Mumford
KIL	0.17	0.68	1224.6	1459.8
STA	7.55	0.74	73926.3	1524.5
COH	0.04	0.26	3.8	0.004

Table 2: Performance on non-cubical data.

Acknowledgement

The authors thank Dmitriy Morozov for helpful discussion and answering our questions concerning Dionysus in depth. We thank Herbert Edelsbrunner for helpful comments.

The first author's work is partially supported by the Austrian Science Fund under grant P20134-N13.

References

[1] P. Bendich, H. Edelsbrunner, and M. Kerber. Computing robustness and peristence for images. *IEEE Transactions on Visualization and Computer Graphics*, 16:1251–1260, 2010.

[2] V. de Silva, D. Morozov, and M. Vejdemo-Johansson. Dualities in persistent (co)homology. Manuscript, 2010.

[3] H. Edelsbrunner and J. Harer. *Computational Topology, An Introduction*. American Mathematical Society, 2010.

[4] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[5] H. Freudenthal. Simplizialzerlegungen von beschränkter Flachheit. *Annals of Mathematics*, 43(3):580–582, 1942.

[6] D. Morozov. Persistence algorithm takes cubic time in the worst case. In *BioGeometry News*. Duke Computer Science, Durham, NC, 2005.

[7] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

[1] `http://www.volvis.org/`

[2] `http://www.mrzv.org/software/dionysus/`

Kinetic Convex Hulls in the Black-Box Model*

Mark de Berg† Marcel Roeloffzen† Bettina Speckmann†

Abstract

Over the past decade, the kinetic-data-structures framework has become the standard in computational geometry for dealing with moving objects. A fundamental assumption underlying the framework is that the motions of the objects are known in advance. This assumption severely limits the applicability of KDSs. We study KDSs in the *black-box model*, which is a hybrid of the KDS model and the traditional time-slicing approach. In this more practical model we receive the position of each object at regular time steps and we have an upper bound on $d_{\max}$, the maximum displacement of any point in one time step.

We study the maintenance of the convex hull of a planar point set P in the black-box model, under the following assumption on $d_{\max}$: there is some constant k such that for any point $p \in P$ the disk of radius $d_{\max}$ contains at most k points. We analyze our algorithms in terms of Δ_k, the so-called *k-spread* of P. We show how to update the convex hull at each time step in $O(k\Delta_k \log^2 n)$ amortized time.

1 Introduction

Motivation. Algorithms dealing with objects in motion traditionally discretize time and recompute the structure of interest at every time step from scratch. This can be wasteful, especially if the time steps are small: then the objects will have moved only slightly, and the structure may not have changed at all. Ideally an object gets attention if and only if its new location triggers an actual change in the structure. *Kinetic data structures (KDSs)*, introduced by Basch *et al.* [3], try to do exactly that: they maintain not only the structure itself, but also additional information that helps to find out when and where the structure will undergo a "real" (combinatorial) change. They maintain a collection of simple geometric tests—these are called *certificates*—with the property that as long as these certificates remain valid, the structure of interest does not change combinatorially. Whenever there is an event—that is, a certificate failure—the KDS is updated. See one of the surveys by Guibas [7, 8, 9] for more information and results on KDSs.

A basic assumption in the KDS framework is that the object trajectories are known. This is necessary to be able to compute the failure times of the certificates, which is essential for the event-driven approach taken in the KDS framework. This assumption severely limits the applicability of the framework. When tracking moving objects, for instance, one gets the object locations only at (probably regular) time steps in an online manner—no detailed knowledge of future trajectories is available. The same is true for physical simulations, where successive locations are computed by a numerical integrator. The goal of our paper is to study the kinetic maintenance of a fundamental geometric structure—the convex hull—in a less restrictive setting: instead of assuming knowledge of the trajectories, we assume only that we know upper bounds on the speeds of the objects and that we get their positions at regular time steps.

Related work. We are not the first to observe that the basic assumption in the KDS model is not always valid. Indeed, the need for a hybrid model, which combines ideas from the KDS model with a traditional time-slicing approach, was already noted in the survey by Agarwal *et al.* [1]. Since then there have been several papers in this direction, as discussed next.

Gao *et al.* [6] study spanners for sets of n moving points in a model where one does not know the trajectories in advance but receives only the positions at each time step. They call this the *blackbox replacement model*—we simply call it the *black-box model*—and show how to update the spanner at each time step in $O(n + k \log \alpha)$ time. Here α is the *spread* of the point set, and k is the number of changes to the hierarchical structure defining their spanner.

Mount *et al.* [11] also study the maintenance of geometric structures in a setting where the trajectories are unknown. They separate the concerns of tracking the points and updating the geometric structure into two modules: the motion processor (MP) is responsible for tracking the points, and the incremental motion algorithm (IM) is responsible for maintaining the geometric structure. Mount *et al.* describe a protocol trying to minimize the interaction between the modules, and they prove that under certain conditions their protocol has good competitive ratio. Their approach goes back to the work of Kahan [10] on certain

*M. Roeloffzen was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 600.065.120. B. Speckmann was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.022.707.

†Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands. {mdberg,mroeloff,speckman}@win.tue.nl

kinetic 1-dimensional problems. See also the more recent work by Cho *et al.* [4]. There is also some work on repairing a triangulation after the vertices have moved. Agarwal *et al.* [2] repair arbitrary planar triangulations, whereas Shewchuk [12] looks at $d-$dimensional Delaunay triangulations.

The results above typically express the running time in terms of the number of changes to the structure at hand, without analyzing this number. This is no surprise; without assumptions on the maximum displacements of the points one cannot say much about the number of changes. On the one hand this "abstract" analysis is appealing since it makes the results general, but on the other hand it becomes hard to decide whether it is better to use these kinetic algorithms or to recompute the structure from scratch at each time step. This is the goal of our paper: to develop KDSs in the black-box model that are provably more efficient than recomputing the structure from scratch under certain assumptions on the trajectories.

Our results. We study black-box KDSs for the convex hull of a set P of n points moving in the plane. As already mentioned, we need to make assumptions on the point movements and time steps to obtain provably efficient solutions. In particular, the time steps should be small enough so that there is some coherence between the positions of the points in consecutive time steps—otherwise we cannot do better than recomputing the structure from scratch. Furthermore, we will assume in most of our results that P is fairly evenly distributed at each time step. We discuss these assumptions in more detail in Section 2.

We present an algorithm that updates the convex hull at each time step in $O(k\Delta_k \log^2 n)$ amortized time, where Δ_k is the k-spread of a point set as defined by Erickson [5]—see Section 2. Note that some proofs are omitted due to space limitations but will be available in the full paper.

2 Preliminaries

Here we introduce some notation, and we discuss some basic issues regarding the black-box model and the concept of k-spread. Although some of our results extend to higher dimensions, we will focus here on the case where P is a set of points moving in the plane.

The black-box model. We denote the position of a point p at time t by $p(t)$, and we let $P(t) := \{p(t) : p \in P\}$ denote the point set at time t. In the black-box model, we assume that we receive the positions at regular time steps $t_0, t_1, \ldots$ and the goal is to update the structure of interest—the convex hull in our case—at each time step. The algorithm need not ask for all new positions at each time step; it may ignore some points if the new locations of these points cannot change the structure. Thus a sublinear update time is potentially feasible and indeed, we show how to obtain sublinear update time for convex-hull maintenance, under certain conditions.

As stated in the introduction, we assume the sampling rate is such that the points in P do not move too much in one time step, as compared to their inter-distances. For a point $p \in P$, let $\text{NN}_k(p, P)$ denote the k-th nearest neighbor of p in $P \setminus \{p\}$. Let $\text{dist}(p, q)$ denote the Euclidean distance between two points p and q, and define $\text{mindist}_k(P) := \min_{p\in P} \text{dist}(p, \text{NN}_k(p, P))$. We assume the sampling rate satisfies the following assumption.

> *Displacement Assumption*: There is a maximum displacement $d_{\max}$ such that
>
> - $d_{\max} \leqslant \min_{t_i} \text{mindist}_k(P(t_i))$, and
> - $\text{dist}(p(t_i), p(t_{i+1})) \leqslant d_{\max}$ for each $p \in P$ and any time step t_i.

The k-spread of a point set. The k-spread Δ_k of P, is defined as

$$\Delta_k(P) := \text{diam}(P)/\text{mindist}_k(P).$$

The k-spread of a point set can be used to bound the number of points within a region if the diameter of the region is not too large.

Lemma 1 *Let P be a set of points in $\mathbb{R}^2$, and let R be a region in $\mathbb{R}^2$ such that $\text{diam}(R) < \text{diam}(P)/\Delta_k(P)$. Then R contains at most k points from P.*

Proof. Assume for contradiction that there are $k+1$ points inside R. Let $p \in P \cap R$. Then $\text{dist}(p, \text{NN}_k(p, P)) \leqslant \text{diam}(R)$. Hence,

$$\Delta_k(P) \geqslant \frac{\text{diam}(P)}{\text{diam}(R)} > \frac{\text{diam}(P)}{\text{diam}(P)/\Delta_k(P)} = \Delta_k(P),$$

a contradiction. □

3 Maintaining the convex hull

Let $\mathcal{CH}(P)$ denote the convex hull of a point set P, and let $\partial\mathcal{CH}(P)$ denote the boundary of $\mathcal{CH}(P)$. In this section we give algorithms to maintain $\mathcal{CH}(P(t))$. From now on, we will use $\mathcal{CH}(t)$ as a shorthand for $\mathcal{CH}(P(t))$. Our algorithms rely on the following observation, which follows from the fact that the distance between p and $\partial\mathcal{CH}(P)$ can change by only $2d_{\max}$ in a single time step.

Lemma 2 *Consider a point $p \in P$, and let $d_p(t) := \text{dist}(p(t), \partial\mathcal{CH}(t))$. Then p cannot become a vertex of $\mathcal{CH}(P)$ until at least $\frac{d_p(t)}{2d_{\max}}$ time steps have passed.*

Lemma 2 suggests the following simple scheme to maintain $\mathcal{CH}(P)$. Compute the initial convex hull

$\mathcal{CH}(t_0)$, and compute for each point $p \in P$ its distance to $\partial\mathcal{CH}(t_0)$. Using this distance and Lemma 2 compute a *time stamp* $t(p)$ for each point p, which is the first time step when p could become a convex-hull vertex. Thus p can be ignored until its time stamp *expires*, that is, until time $t(p)$. In a generic time step t_i, we now determine the set $Q(t_i)$ of all points whose time stamps expire, compute their convex hull and compute new time stamps for the points in $Q(t_i)$. We may use $\mathcal{CH}(t_{i-1})$ to compute the new convex hull, but we don't need to here.

To implement this algorithm we use an array A where $A[t_i]$ contains the points whose time stamps expire at time t_i. To restrict the amount of storage we use an array $A[0..n-1]$ with n entries, and we let time advance through the array in a cyclic manner (using without loss of generality that $t_i = i$). Furthermore, we bound the time stamps to be at most n steps, and we use an approximation of $\mathrm{dist}(p(t), \partial\mathcal{CH}(t))$ to speed up the computations. Our approach is made explicit in Algorithm 1. Note that the algorithm needs to know only $d_{\max}$ to work correctly, it does not need to know bounds on the k-spread.

Algorithm 1: UPDATECH

1 $Q(t) \leftarrow$ set of points stored in $A[t]$
2 Compute $\mathcal{CH}(Q(t))$ and set $\mathcal{CH}(t) \leftarrow \mathcal{CH}(Q(t))$.
3 **for** each $p \in Q(t)$ **do**
4 $\quad d_p^* \leftarrow$ lower bound on $\mathrm{dist}(p(t), \partial\mathcal{CH}(t))$.
5 $\quad$ Set p's time stamp:
$\quad t(p) \leftarrow [t + \min(1 + \lfloor \frac{d_p^*}{2d_{\max}} \rfloor, n)] \mod n$.
6 $\quad$ Add p to $A[t(p)]$.
7 $t \leftarrow (t+1) \mod n$

It remains to describe how to compute $\mathcal{CH}(Q(t))$ in Step 2 and how to compute the values d_p^* in Step 5. Computing $\mathcal{CH}(Q(t))$ can be done by an optimal convex-hull algorithm in $O(|Q(t)| \log |Q(t)|)$ time. To compute d_p^* we proceed as follows. Let q_{above} be the point on $\partial\mathcal{CH}(t)$ directly above p, and define q_{below}, q_{left}, and q_{right} similarly. These points can be found in logarithmic time using binary search. Let $q_{\min}$ denote the minimum distance between p and any of the points q_{above}, q_{below}, q_{left}, and q_{right}. Then we set $d_p^* = q_{\min}/\sqrt{2}$ (Note that $d_p^* \leqslant \mathrm{dist}(p, \partial\mathcal{CH}(t)) \leqslant \sqrt{2}\ d_p^*$.) We get the following result.

Lemma 3 *At each time step t,* UPDATECH *updates the convex hull in $O(|Q(t)| \log n)$ time.*

Next we analyse $|Q(t)|$, the number of time stamps that can expire in a single time step.

Analyzing the number of expiring time stamps. We perform our analysis in terms of Δ_k, which is an upper bound on the k-spread of P at any time. We first bound the number of convex hull vertices.

Lemma 4 *The number of vertices of the convex hull $\mathcal{CH}(P)$ of a point set P is $O(k\Delta_k(P))$.*

Proof. The length of $\partial\mathcal{CH}(P)$ is $\Theta(\mathrm{diam}(P))$, so we can cut $\partial\mathcal{CH}(P)$ into $\Theta(\mathrm{diam}(P)/\mathrm{mindist}_k(P)) = \Theta(\Delta_k(P))$ pieces with a length less then $\mathrm{mindist}_k(P)$. From Lemma 1 we know that each such piece contains at most k points. It follows that $\partial\mathcal{CH}(P)$ contains $O(k\Delta_k(P))$ vertices. □

Now let us consider the number of expiring time stamps. In the worst case it can happen that all time stamps expire in a single time step. However, using an amortization argument we show that on average only $O(k\Delta_k \log n)$ time stamps expire in each time step.

Lemma 5 *The amortized number of time stamps expiring in each time step is $O(k\Delta_k \log n)$.*

Proof. (Sketch) We prove the lemma using the accounting method: each point has an account into which we put a certain amount of money at each time step, and whenever the time stamp of a point expires it has to pay 1 euro from its account. Our scheme is that at time step t_i each point p receives

$$\min\left(1,\ \max\left(\frac{1}{n}, \frac{8\sqrt{2}\cdot d_{\max}}{d_p(t_i)}\right)\right) \text{ euro},$$

where $d_p(t_i) = \mathrm{dist}(p(t_i), \partial\mathcal{CH}(t_i))$.

To prove each point can pay 1 euro when it expires, consider a point p whose time stamp expires at time t_i, and let $t_j < t_i$ be the previous time step when p's time stamp expired. (If there is no such time step, we can take $j = 0$.) Now define $t(p) := t_i - t_j = i - j$ to be the number of time steps from t_j up to t_{i-1}. If $t(p) = n$ then p certainly has enough money in its account at time t_i, so assume this is not the case. Then we can show that $t(p) \geqslant \frac{d_p(t_j)}{2\sqrt{2}\cdot d_{\max}}$.

The distance between $p(t_m)$ and $\mathcal{CH}(t_m)$ for $t_j \leqslant t_m \leqslant t_{i-1}$ is at most $4d_p(t_j)$. Hence point p gets at least $\frac{8\sqrt{2}d_{\max}}{4\cdot d_p(t_j)}$ euro per time step, which proves each point has at least one euro when it expires.

Next we prove that we only spend $O(k\Delta_k \log n)$ euro per time step. We consider the points p such that $d_p(t_i) \leqslant 8\sqrt{2}n \cdot d_{\max}$; the remaining points get $1/n$ euros each, so in total at most 1 euro. We divide these points into groups $G_1, \ldots, G_\ell$ (see Figure 1). Each group G_j contains the points $p \in P$ such that $(j-1)\cdot d_{\max} \leqslant d_p(t_i) \leqslant j \cdot d_{\max}$, where $\ell = 8\sqrt{2}n$. We can show that each group contains $O(k\Delta_k)$ points, so we pay at most $O(k\Delta_k/j)$ euro per group.

Summing this over all groups we see that the amount we pay at each time step is

$$\sum_{j=1}^{8\sqrt{2}n} O\left(\frac{k\Delta_k}{j}\right) = O(k\Delta_k \log n).$$

□

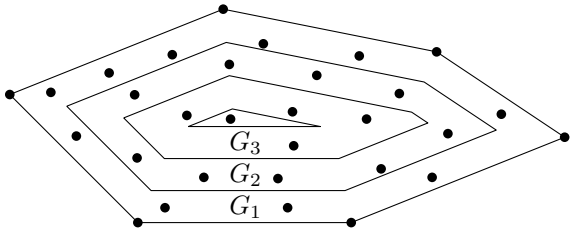


Figure 1: We divide the points into groups based on their distance to $\partial\mathcal{CH}(t_i)$.

From Lemma 3 and 5 we conclude the following:

Theorem 6 *Under the Displacement Assumption, the convex hull of a set P of n points moving in the plane can be maintained in the black-box model in $O(k\Delta_k \log^2 n)$ amortized time per time step, where Δ_k is the maximum k-spread of P at any time.*

4 Conclusion

We presented an algorithm to maintain the convex hull of a planar point set in the KDS black-box model. The algorithm is simple and does not require knowledge of $\Delta_k(P)$ or k: it only needs to know $d_{\max}$, the maximum displacement of any point in one time step.

We spend $O(k\Delta_k \log^2 n)$ amortized time to update the convex hull after each time step. This is optimal up to the logarithmic factors, because the convex hull can undergo $\Omega(k\Delta_k)$ changes in any time step. Moreover, we can show that our bound $O(k\Delta_k \log n)$ on the number of expiring time stamps is tight. However, it may be possible to get rid of one logarithmic factor from the time bound by a more clever algorithm. In fact, if we are allowed to use the floor function, then we know how to to do this. Unfortunately, the resulting algorithm needs to know $\text{mindist}_k(P(t))$, which is perhaps not very realistic. It would be interesting to design an algorithm that needs to know only $d_{\max}$ and achieves $O(k\Delta_k \log n)$ update time. Another interesting open problem is whether it is possible to make the time bound worst-case rather than amortized.

In the full paper we also studied the convex hull problem without a bound on the spread Δ_k. In this case it is still possible to do updates efficiently, namely in $O(n \log k)$ time per time step. Note that when $\Delta_k > n$ all points can appear as vertices on the convex hull giving a lower bound of $\Omega(n)$ on the update time.

Another interesting problem we studied in the KDS black-box model is the Delaunay triangulation. Using the k-spread and displacement assumption we can update the Delaunay triangulation using $O(k^2\Delta_k^2)$ flips in $O(k^2\Delta_k^2 \log n)$ time by moving the points one by one from their old to their new locations. We can reduce the update time to $O(k^2\Delta_k^2)$ by inserting $p(t)$ and removing $p(t-1)$ for each point $p \in P$.

How realistic our model is depends on the application, of course. We expect that in many applications the sampling rate is such that the Displacement Assumption is satisfied. The other question is whether the point set can be expected to have small k-spread. In meshing-type applications, it may be realistic to assume that the k-spread is $O(\sqrt{n})$. In any case, Δ_k seems like a reasonable parameter to measure the efficiency, and we think it will be interesting to study other structures in the KDS black-box model under the Displacement Assumption and to analyze their performance in terms of Δ_k.

References

[1] P.K. Agarwal, L.J. Guibas, H. Edelsbrunner, J. Erickson, M. Isard, S. Har-Peled, J. Hershberger, C. Jensen, L. Kavraki, P. Koehl, M. Lin, D. Manocha, D. Metaxas, B. Mirtich, D. Mount, S. Muthukrishnan, D. Pai, E. Sacks, J. Snoeyink, S. Suri, and O. Wolefson. Algorithmic issues in modelling motion. *ACM Comput. Surv.* 34:550–572 (2002).

[2] P.K. Agarwal, B. Sadri, and H. Yu. Untangling triangulations through local explorations. In *Proc. 24th ACM Sympos. Comput. Geom.*, pages 288–297, 2008.

[3] J. Basch, L.J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discr. Algorithms*, pages 747–756, 1997.

[4] M. Cho, D.M. Mount, and E. Park. Maintaining nets and net trees under incremental motion. In *Proc. 20th Int. Sympos. Algo.Comput.*, pages 1134–1143, 2009.

[5] J. Erickson. Dense Point Sets Have Sparse Delaunay Triangulations. In *Discrete and Computational Geometry* 30:83-115 (2005).

[6] J. Gao, L.J. Guibas, A. Nguyen. Deformable spanners and applications. In *Proc. 20th ACM ACM Sympos. Comput. Geom.*, pages 190–199, 2004.

[7] L.J. Guibas. Kinetic data structures—a state-of-the-art report. In *Proc. 3rd Workshop Algorithmic Found. Robot.*, pages 191–209, 1998.

[8] L.J. Guibas. Kinetic data structures. In: D. Mehta and S. Sahni (editors), *Handbook of Data Structures and Applications*, Chapman and Hall/CRC, 2004.

[9] L.J. Guibas. Motion. In: J. Goodman and J. O'Rourke (eds.), *Handbook of Discrete and Computational Geometry (2nd edition)*, pages 1117–1134. CRC Press, 2004.

[10] S. Kahan. A model for data in motion. In *Proc. 23rd ACM Sympos. Theory Comput.*, pages 267–277, 1991.

[11] D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. A computational framework for incremental motion. In *Proc. 20th ACM Sympos. Comput. Geom.*, pages 200–209, 2004.

[12] R. Shewchuk. Star splaying: an algorithm for repairing Delaunay triangulations and convex hulls. In *Proc. 21st ACM Sympos. Comput. Geom.*, pages 237–246, 2005.

Boundary of a non-uniform point cloud for reconstruction

Nicolas Chevallier* Yvan Maillot†

Abstract

This paper deals with the problem of shape reconstruction. We define a filtration of the Delaunay complex of a point cloud. This filtration allows to select points in a point cloud that should be boundary points. Theoretical guarantees are given when the point cloud samples a region with smooth boundary. A simple and efficient algorithm computing the filtration is described.

1 Introduction

We will focus on the reconstruction of an open set Ω in $\mathbb{R}^d$ with smooth boundary, from a set of points non-uniformly distributed inside Ω and not just lying on its boundary.

With our hypothesis, $\overline{\Omega}$, the closure of Ω, is a manifold with boundary. In the past decade, many algorithms have been proposed for the reconstruction of manifolds [2, 3, 4]. Some of them give guarantees, but only for surfaces without boundary. Recently, in [7], Dey et al. have been the firsts to ensure the reconstruction of surfaces with boundary. Nevertheless, their algorithm is limited to 2-dimensional surfaces in $\mathbb{R}^3$, from a sufficiently dense sample. Besides, their algorithm faces problems when the sample is not uniform and no theoretical guarantees are provided in this case.

Given a point cloud S sampling a bounded open set Ω in $\mathbb{R}^d$ with smooth boundary, our work proposes:

- To define a filtration of the Delaunay complex of S. This filtration takes into account in a very natural way that the local density of the cloud may change according to the local complexity of Ω. This filtration depending on a positive real number α is called *Locally-Density-Adaptative-α-complex, LDA-α-complex* for short.
- To select a subset in the point cloud S, called *LDA-α-boundary*, that presumably constitutes a sample of the boundary F of Ω.
- Thanks to a result of Chazal and Lieutier [6] to prove that the LDA-α-boundary carries topological and geometric properties of F.
- To give an efficient algorithm for computing LDA-α-complex and LDA-α-boundary.

*UHA, LMIA/SD, Mulhouse, France, nicolas.chevallier@uha.fr
†UHA, LMIA/MAGE, Mulhouse, France, yvan.maillot@uha.fr

2 Mathematical and geometric concepts

Notations. $d(x, y) = \|x - y\|$ denotes the Euclidean distance between to points x, y in $\mathbb{R}^d$.

$B(x, r) = \{y \in \mathbb{R}^d, d(x, y) \leq r\}$ denotes the closed ball of center x and radius $r \in [0, \infty[$.

Let A be a subset of $\mathbb{R}^d$, A^C denotes its complementary in $\mathbb{R}^d$, $\overline{A}$ its closure, A^o its interior, ∂A its boundary, $\text{conv}(A)$ its convex hull, and $\text{diam}\, A$ its diameter, i.e. $\text{diam}\, A = \sup\{d(x, y) : x, y \in A\}$.

Delaunay complex. Let S be a finite point set in $\mathbb{R}^d$. An *empty ball* is a ball containing no point of S in its interior.

The *Delaunay complex* of S, $\text{Del}(S)$, is the set of convex polytopes $K = \text{conv}(S \cap B)$ where B is an empty ball.

Maximal ball. A ball B of $\mathbb{R}^d$ is called a *maximal ball* if B is an empty ball such that the dimension of the convex polytope $\text{conv}(B \cap S)$ is d.

Restricted Delaunay complex. Let X be a point set in $\mathbb{R}^d$. A convex polytope K in the Delaunay complex $\text{Del}(S)$ is in the Delaunay complex of S restricted to X, denoted by $Del_{|X}(S)$, if there exists an empty ball $B = B(x, r)$ such that $x \in X$ and the vertices of K are on the boundary of B. This means that K is a face of $\text{conv}(S \cap B)$. The set $Del_{|X}(S)$ of all these convex polytopes K is a sub-complex of $\text{Del}(S)$.

We assume in the following that S is a finite point set in $\mathbb{R}^d$ which is contained in no proper affine subspace of $\mathbb{R}^d$.

3 Idea of sampling and reconstruction

We aim at reconstructing a bounded open set Ω with smooth boundary from a sample set S included in $\overline{\Omega}$ and not only on its boundary. Moreover, the sample does not have to be uniformly distributed, i.e., its local density may be different in places. Nevertheless, the information about the shape to reconstruct has to be carried by the sample. The sample has to be dense where a big amount of details is required and a sudden variation of local density indicates the presence of a hole or a hollow, as shown on figure 1 close to the eyes of the lizard or on its fingers. Since variations of the local density are allowed, the sample may be

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

less dense where the shape is simple. However, the density must change gradually, as shown on figure 1 on the back of the lizard, to avoid the formation of non-existent holes.

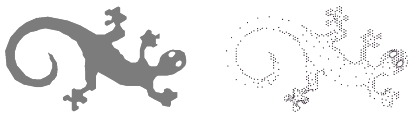
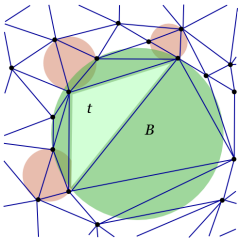


Figure 1: A shape (left), a possible sample (right)

The problem comes down to finding an efficient way to measure the variation of local density of the sample set S.

4 Formalization of the idea

The Delaunay complex can be efficiently used to measure the density variation of the sample set S. For instance, in the figure opposite, the maximal ball B is much larger than some of its neighbours. This means there is a wide area with no point of S while the sampling around is dense. This area is probably a hole and the polytope t included in B must be eliminated.



4.1 Eliminating polytopes

Polytopes will be eliminated by eliminator balls. An eliminator ball is an empty ball large enough compared to the maximal balls around.

Definition 1 *Let α be a real number in $]0,\infty[$. An empty ball $B = B(c,r)$ is* α-eliminator *if for each point p of S lying on ∂B, there exists a maximal ball $B' = B'(c',r')$ with p lying on $\partial B'$ such that $r \geq \alpha r'$.*

Here are some interesting observations:

- There exists an α_∞ such that $\forall \alpha \geq \alpha_\infty$ there is no α-eliminator ball since the radius of a maximal ball is strictly positive and the radius of an empty ball is finite.
- There exists an α_0 such that $\forall \alpha \leq \alpha_0$ all empty balls containing a 1-polytope are α-eliminator since the radius of an empty ball containing a 1-polytope is strictly positive and the radius of a maximal ball is finite.
- A maximal ball $B = B(c,r)$ is 1-eliminator since the ball $B'(c',r')$ with p lying on $\partial B'$ such that $r \geq r'$ is B itself.

Definition 2 *A polytope K of* $\mathrm{Del}(S)$ *is* α-eliminated *if each empty ball B such that $K \subseteq \mathrm{conv}(S \cap \partial B)$ is α-eliminator.*

The following observations are deduced from the previous ones: No polytope is α_∞-eliminated. All the polytopes except the 0-polytopes are α_0-eliminated. All d-polytopes are 1-eliminated.

4.2 LDA-α-complex

Definition 3 *The* LDA-α-complex *is the set of all polytopes K of* $\mathrm{Del}(S)$ *that are not* α-eliminated.

We can collect some easy facts: The LDA-α-complex of S is a sub-complex of $\mathrm{Del}(S)$. The LDA-α_0-complex of S is S. The LDA-α_∞-complex of S is $\mathrm{Del}(S)$. The LDA-1-complex of S is a sub-complex of $\mathrm{Del}(S)$ with no d-polytope. If $\alpha_1 \leq \alpha_2$ then LDA-α_1-complex $\subseteq$ LDA-α_2-complex.

The LDA-α-complex is indeed very close to the conformal-α-shape [5] with α_p^- equals to 0 and α_p^+ equals to the radius of the smallest maximal ball containing p.

4.3 LDA-α-boundary

Definition 4 *The* LDA-α-boundary *is the union of the sets of the vertices of all the α-eliminated polytopes and of the set of all the vertices of the convex hull of S.*

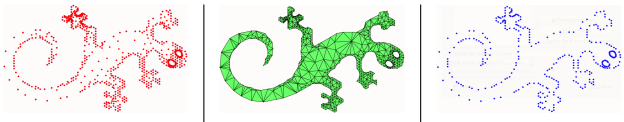
5 Algorithm

We present a very simple and efficient algorithm in the full version of this paper. A short description is given in what follows.

5.1 Description

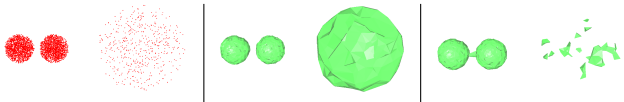
There are four main steps. First, $\mathrm{Del}(S)$ is computed. The second step computes for each $v \in S$, the radius $m(v)$ of the smallest maximal ball containing v .The third step computes for each polytope K in $\mathrm{Del}(S)$, the radius $e(K)$ of the smallest empty ball containing K. The fourth step determines the polytopes in $\mathrm{Del}(S)$ that are in the LDA-α-complex.

The fourth step selects the polytopes of $\mathrm{Del}(S)$ in decreasing order of dimensions. First, for each d-polytope K, if the inequality $e(K) < \alpha m(v)$ holds for at least one vertex v of K, then K is in the LDA-α-complex. Moreover, the LDA-α-boundary is determined at the same time since the elements of the LDA-α-boundary are the vertices of the d-polytopes that are not selected (together with the vertices of the convex hull of S). Next, once the polytopes of dimensions $\geq k$ have been selected, a $(k-1)$-polytope K of $\mathrm{Del}(S)$ is in the LDA-α-complex if either one of the



A sample of a lizard | Its LDA-2 -complexe | Its LDA-2 -boundary

Figure 2: The lizard



Three balls sampled (a) | with LDA-α-shapes (b) | with α-shapes (c)

Figure 3: The balls

k-polytope containing K is in the LDA-α-complex or if $e(K) < \alpha m(v)$ for at least one vertex v of K.

In the full version of this paper we prove that the time-complexity of the algorithm is $O(|\mathrm{Del}(S)|)$ plus the time required to compute the Delaunay complex $\mathrm{Del}(S)$.

5.2 Little examples

Thanks to CGAL [1], this algorithm has been implemented in 2D and 3D. Two examples are shown in this subsection. Figure 2 shows the sample of a lizard, its LDA-2-complex, and its LDA-2-boundary. The density of this sample is strong only where it is needed. Figure 3 (a) shows 3D-balls that have been randomly sampled according to non-constant probability densities. The reconstructions work with LDA-α-complexes while it is well known that α-shapes may fail in that case.

6 Well distributed subsets

Given a bounded open set Ω with smooth boundary and a finite point set S in $\overline{\Omega}$, we will give some conditions about S ensuring that with a right choice of α, the LDA-α-boundary of S carries the topological informations about $\partial\Omega$.

We first define (ε, δ)-sample with a general lipschitz function. We will specify such a function later.

Definition 5 *Let $f : \mathbb{R}^d \to \mathbb{R}_+$ be a k-lipschitz function. Let δ and ε be two positive real numbers.*

A point set S is an (ε, δ)-sample of $\overline{\Omega}$ if $S \subset \overline{\Omega}$, and if

Density: $\forall x \in \overline{\Omega},\ \exists y \in S,\ d(x, y) < \varepsilon f(x)$,

Sparsity: $\forall x, y \in S,\ x \neq y \Rightarrow d(x, y) \geq \delta f(x)$.

In the two following lemmas, we assume that $\overline{\Omega}$ is (ε, δ)-sampled and that f is k-lipschitz.

7 Two technical lemmas

We want to provide theoretical guarantees that the LDA-α-boundary is a good approximation of the boundary F of Ω.

To do this we have to prove these two results: (1) Any point in the LDA-α-boundary is near F. (2) Any point in F is near a point of the LDA-α-boundary.

Next lemma shows that with a right choice of the parameters $\alpha, \varepsilon, \delta$, the LDA-$\alpha$-boundary is close to F.

Lemma 6 ***(elimination lemma).*** *Suppose $\varepsilon < \frac{1}{2k}$ and $\alpha > \frac{2\varepsilon}{\delta(1-2k\varepsilon)}$. If p is a vertex of a LDA-α-eliminated polytope t_p, then*

$$d(p, F) \leq \frac{\varepsilon}{1 - k\varepsilon} f(p) \tag{1}$$

$$d(c, \overline{\Omega}) > 0 \tag{2}$$

where c is the center of any empty ball B such that $t_p \subset \mathrm{conv}(S \cap B)$.

7.1 Statements of main results about (ε, δ)-samples

The first inequality in the elimination lemma shows immediately that points in the LDA-α-boundary are close to F:

Corollary 7 ***(α-boundary).*** *Suppose $\alpha > \frac{2\varepsilon}{\delta(1-2k\varepsilon)}$ and $\varepsilon < \frac{1}{2k}$. Then all $p \in S$ that are in the LDA-α-boundary of S, are at a distance from F lower or equal to $\frac{\varepsilon}{1-k\varepsilon} f(p)$.*

The second inequality in the elimination lemma immediately leads to an information about $\mathrm{Del}_{|\overline{\Omega}}(S)$ the restricted Delaunay triangulation of S with respect to $\overline{\Omega}$:

Corollary 8 *Suppose $\alpha > \frac{2\varepsilon}{\delta(1-2k\varepsilon)}$ and $\varepsilon < \frac{1}{2k}$. Then the restricted Delaunay complex $\mathrm{Del}_{|\overline{\Omega}}(S)$ is a subcomplex of the LDA-α-complex.*

Until now, we only assume the function f to be k-lipschitz. In order to go further, we need to specify the function f. Let $\mathcal{S}_F$ be the *skeleton* of F, that is the set of centers of maximal open balls that do not meet F. In the following $f : \mathbb{R}^d \to \mathbb{R}$ is defined by $f(x) = d(x, F) + d(x, \mathcal{S}_F)$. It is a 2-lipschitz function.

With this more precise assumption about the sample set S, we are not able to prove the reverse inclusion LDA-α-complex $\subset \mathrm{Del}_{|\overline{\Omega}}(S)$. Nevertheless, we can prove that the LDA-α-complex is included in the Delaunay complex of S restricted to a neighborhood of $\overline{\Omega}$.

Proposition 9 ***(neighborhood).*** *Suppose that $\varepsilon < \frac{1}{20}$ and $\frac{1}{4\varepsilon} > \alpha > 2$. Then a maximal ball $B(c, R)$ such that $d(c, \overline{\Omega}) \geq \frac{2\varepsilon\alpha}{1-4\varepsilon\alpha} f(c)$ is α-eliminator.*

By the first corollary, any point in the LDA-α-boundary is close to F. Conversely any point in F is close to a point of the LDA-α-boundary of S:

Proposition 10 *(α-**boundary**). Suppose that $\varepsilon < \frac{1}{20}$ and $\frac{1}{32\varepsilon} > \alpha > 2$. Then for any p in F there exists a point p_0 in the LDA-α-boundary of S whose distance to p is $\leq 8\alpha\varepsilon f(p)$.*

8 A reconstruction result

The last step is to see that with a good choice of α, ε, and δ, the topological information about F is contained in the α-boundary. For this, we will use a result of F. Chazal and A. Lieutier [6]. Their result allows a reconstruction of a compact manifold Σ of dimension $d-1$ starting with a compact set K close to Σ. They have proved that if K is close enough to Σ, then the boundary of a union of balls $B(p, r_p)$ centered at the points p in K, is the union of two subsets homeomorphic to Σ. This shows that K carries the topological informations about Σ.

Notations. Let Σ be a compact set in $\mathbb{R}^d$. For x in $\mathbb{R}^d$, denote by $\pi(x)$ a point in Σ nearest to x. This point is unique if x is not in the medial axis of Σ. Denote by $LFS(x)$ the distance from x to the skeleton of Σ.

Definition 11 *Let κ and ρ be positive real numbers, and let Σ be a compact manifold in $\mathbb{R}^d$. A compact set $K \subset \mathbb{R}^d$ is a (κ, ρ)-approximation of Σ if :*
i. For all $p \in K$, $d(p, \pi(p)) \leq \kappa\rho LFS(p)$,
ii. For all $p \in \Sigma$, there exists a point $q \in K$ such that $d(p, \pi(q)) < \rho LFS(p)$.

Theorem 12 *(Chazal, Lieutier) Let κ, ρ, and $0 < a < b < \frac{1}{3} - \kappa\rho$ be such that*

$$(1-a')^2 + \left((b'-a') + \frac{b(1+2b'-a')}{1-b-\kappa\rho}\right)^2 < \left(1 - \frac{\kappa\rho(1+2b'-a')}{1-b-\rho}\right)^2$$

with $a' = (a-\kappa\rho)(1-\rho) - \rho$ and $b' = \frac{b+\kappa\rho}{1-2(b+\kappa\rho)}$. Let K be a (κ, ρ)-approximation of Σ. Let $(r_p)_{p\in K}$ be a family of real numbers such that $a \leq \frac{r_p}{LFS(p)} \leq b$ for all $p \in K$ and set

$$\mathcal{K} = \mathcal{K}((r_p)_{p\in K}) = \bigcup_{p\in K} B(p, r_p).$$

Then
- Σ is a deformation retract of $\mathcal{K}$.
- $\mathcal{K}$ is homeomorphic to any tubular neighborhood $\{x \in \mathbf{R}^d : d(x,\Sigma) \leq s\}$ where $s < reach(\Sigma) = d(\Sigma, \mathcal{M}_\Sigma)$.
- The boundary $\partial\mathcal{K}$ is an isotopic hypersurface to $\Sigma_s = \{x \in \mathbb{R}^d : d(x,\Sigma) = s\}$.

Remark 1. If Σ is an hypersurface, $\partial\mathcal{K}$ is isotopic to 2 copies of Σ (isotopy$\Longrightarrow$ homeorphism).

In order to reconstruct the boundary $F = \partial\Omega$ starting with the (ε, δ)-sample set S, it is enough to use this theorem with $\Sigma = F$ and K the LDA-α-boundary of S.

Proposition 13 *Choose $\alpha = 10$, $\varepsilon \leq \frac{1}{2500}$, and $\delta = \frac{\varepsilon}{4}$. Then Chazal and Lieutier's theorem works with $\Sigma = F$ and $K =$ the LDA-α-boundary of S*

All the proofs are in the full version of this paper.

References

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry*, 22:481–504, 1999.

[3] N. Amenta, S. Choi, T. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *ACM Symposium on Computational Geometry*, pages 213–222, 2000.

[4] J.-D. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proceedings of the sixteenth annual symposium on Computational geometry*, SCG '00, pages 223–232, New York, NY, USA, 2000. ACM.

[5] F. Cazals, J. Giesen, M. Pauly, and A. Zomorodian. The conformal alpha shape filtration. *The Visual Computer*, 22(8):531–540, 2006.

[6] F. Chazal and A. Lieutier. Smooth manifold reconstruction from noisy and non-uniform approximation with guarantees. *Computational Geometry*, 40(2):156 – 170, 2008.

[7] T. K. Dey, K. Li, E. A. Ramos, and R. Wenger. Isotopic reconstruction of surfaces with boundaries. *Comput. Graph. Forum*, 28(5):1371–1382, 2009.

Convex Hull of Imprecise Points in $o(n \log n)$ Time after Preprocessing

Esther Ezra* Wolfgang Mulzer†

Abstract

Motivated by the desire to cope with *data imprecision* [8], we study methods for preprocessing a set of planar regions such that whenever we are given a set of points, each of which lies on a distinct region, we can compute a specified structure on these points more efficiently than in "standard settings" (that is, without preprocessing).

In particular, we study the following problem. Given a set L of n lines in the plane, we wish to preprocess L such that later, upon receiving a set P of n points, each of which lies on a distinct line of L, we can construct the convex hull of P efficiently. We show that in quadratic time and space it is possible to construct a data structure on L that enables us to compute the convex hull of any such point set P in $O(n\alpha(n) \log^* n)$ expected time. The analysis applies almost verbatim when L is a set of line-segments, and yields the same asymptotic bounds.

1 Introduction

Most studies in computational geometry rely on an unspoken assumption: whenever we are given a set of input points, their precise locations are available to us. Nowadays, however, the input is often obtained via sensors from the real world, and hence it comes with an inherent imprecision. Accordingly, an increasing effort is being devoted to achieving a better understanding of data imprecision and to developing tools to cope with it (see, e.g., [8] and the references therein). The notion of imprecise data can be formalized in numerous ways [8]. We consider a particular setting that has recently attracted considerable attention (see [2] and the references therein). We are given a set of planar regions, each of which represents an estimate about an input point, and the exact coordinates of the points arrive some time later and need to be processed quickly. This situation could occur, e.g., during a two-phase measuring process: first the sensors quickly obtain a rough estimate of the data, and then they invest considerably more time to find the precise locations. This raises the necessity to preprocess the preliminary (imprecise) locations of the points, and store them in an appropriate data structure, so that when the exact measurements of the points arrive we can efficiently compute a prespecified structure on them. In settings of this kind, we assume that for each input point its corresponding region is known (note that by this assumption we also avoid a point-location overhead). In light of the applications, this is a reasonable assumption, and it can be implemented by, e.g., encoding this information in the ordering of P.

Data imprecision. Previous work has mainly focused on computing a triangulation for the input points. Held and Mitchell [5] were the first to consider this framework, and they obtained optimal bounds for preprocessing disjoint unit disks for point set triangulations, a result that was later generalized by van Kreveld *et al.* [7] to arbitrary disjoint polygonal regions. For *Delaunay* triangulations, Löffler and Snoeyink [10] obtained an optimal result for disjoint unit disks (see also [4, 9]), which was later simplified and generalized by Buchin *et al.* [2] to *fat*[1] and possibly intersecting regions. The preprocessing phase typically takes $O(n \log n)$ time and results in a linear size data structure; the time for finding the structure on the exact point set is usually linear or depends on the complexity (and the fatness) of the input regions.

Since the convex hull can be easily extracted from the Delaunay triangulation in linear time, the same bounds carry over. However, once the regions are not necessarily fat, the techniques in [2, 10] do not yield the aforementioned bounds anymore. In particular, if the regions consist of lines or line-segments, one cannot hope (under certain computational models) to construct the Delaunay triangulation of P in time $o(n \log n)$, regardless of preprocessing (see [10]). Nevertheless, if we are less ambitious and just wish to compute the *convex hull* of P, we can achieve better performance, which is the main problem studied in this paper.

Convex hull. Computing the convex hull of a planar n-point set is perhaps the most fundamental problem in computational geometry, and there are many algorithms available [1]. All these algorithms require $\Theta(n \log n)$ steps, which is optimal in the algebraic computation tree model. However, there are numerous ways to exploit additional information to improve

*Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA; esther@cims.nyu.edu.

†Institut für Informatik, Freie Universität Berlin, 14195 Berlin, Germany; mulzer@inf.fu-berlin.de.

[1] A planar region o is said to be fat if there exist two concentric disks, $D \subseteq o \subseteq D'$, such that the ratio between the radii of D' and D is bounded by some constant.

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

this bound. For example, if the points are sorted along any fixed direction, Graham's scan takes only linear time [1]. If we know that there are only h points on the hull, the running time reduces to $O(n \log h)$ [6]. Our work shows another setting in which additional information can be used to circumvent the theoretic lower bound.

Our results. We show that we can preprocess the input lines L such that given any set P of points, each of which lies on a distinct line of L, the convex hull $\mathrm{CH}(P)$ can be computed in expected time $O(n\alpha(n) \log^* n)$, where $\alpha(\cdot)$ is the (slowly growing) inverse Ackermann function [12, Chapter 2.1]. Our data structure has quadratic preprocessing time and storage, and the convex hull algorithm is based on a batched randomized incremental construction similar to Seidel's tracing technique [11]. As part of the construction, we repeatedly trace the *zone* of (the boundary of) an intermediate hull in the *arrangement* of the input lines.[2] The fact that the complexity of the zone is only $O(n\alpha(n))$ [12], and that it can be computed in the same asymptotic time bound (after having the arrangement at hand), is a key property of our solution. The analysis applies almost verbatim when L is a set of line-segments, and we obtain similar asymptotic bounds.

2 Convex Hulls

Preliminaries The input at the preprocessing stage is a set L of n lines in the plane. A *query* to the resulting data structure consists of any point set P such that each point lies on a distinct line in L, and for every point we are given its corresponding line. For simplicity, and without loss of generality, we assume that both L and P are in *general position* (see, e.g., [1, 12]). We denote by $\mathrm{CH}(P)$ the convex hull of P, and by $E(P)$ the edges of $\mathrm{CH}(P)$. We represent the vertices of $\mathrm{CH}(P)$ in clockwise order, and we direct each edge $e \in E(P)$ such that $\mathrm{CH}(P)$ lies to its right. Given a subset $Q \subset P$, a point $p \in P \setminus Q$, and an edge $e \in E(Q)$, we say that e is in *conflict* with p if p lies to the left of the line supported by e. The set of all points in $P \setminus Q$ in conflict with e is called the *conflict list* C_e of e, and the size of C_e is called the *conflict size* c_e of e.

2.1 The Construction

Preprocessing. We construct in $O(n^2)$ time (and storage) the *arrangement* $\mathcal{A}(L)$ of L, and produce its *vertical decomposition*, that is, we erect an upward and a downward vertical rays through each vertex v of $\mathcal{A}(L)$ until they meet some line of L (not defining v), or else extend to ∞. The *complexity* of a face f in $\mathcal{A}(L)$ is the number of edges incident to f. The *zone* of a curve γ consists of all faces that intersect γ, and the complexity of the zone is the sum of their complexities.

Queries. Given an exact point set $P = \{p_1, \ldots, p_n\}$ as described above, we obtain $\mathrm{CH}(P)$ through a *batched* randomized incremental construction as follows: Let $P_1 \subseteq P_2 \subseteq \cdots \subseteq P_{\log^* n} = P$ be a sequence of subsets, where P_{k-1} is a random sample of P_k of size $z_{k-1} := \min\{\lfloor n/\log^{(k-1)} n \rfloor, n\}$, for $k = 2, \ldots, \log^* n$. Here, $\log^{(i)} n$ is the ith iterated logarithm: $\log^{(0)} n = n$ and $\log^{(k)} n = \log(\log^{(k-1)} n)$. This sequence of subsets is called a *gradation*. The idea is to construct $\mathrm{CH}(P_1)$, $\mathrm{CH}(P_2)$, ..., $\mathrm{CH}(P_{\log^* n})$ one by one. First, we have $|P_1| = O(n/\log n)$, so it takes $O(n)$ time to find $\mathrm{CH}(P_1)$, using, e.g., Graham's scan [1]. Then, for $k = 2, \ldots, \log^* n$, we incrementally construct $\mathrm{CH}(P_k)$ by updating $\mathrm{CH}(P_{k-1})$. This basic technique was introduced by Seidel [11] and it has later found many more applications.

To construct $\mathrm{CH}(P_k)$ from $\mathrm{CH}(P_{k-1})$, we use the data structure from the preprocessing to quickly construct the conflict lists of the edges in $E(P_{k-1})$ with respect to P_k. In the standard Clarkson-Shor randomized incremental construction [3] it takes $O(n \log n)$ time to maintain the conflict lists. However, once we have the arrangement $\mathcal{A}(L)$ at hand, this can be done significantly faster. In fact, we use a refinement of the conflict lists: we shoot an upward vertical ray from each point on the upper hull of P_{k-1}, and a downward vertical ray from each point on the lower hull. Furthermore, we erect vertical walls through the leftmost and the rightmost points of $\mathrm{CH}(P_{k-1})$. This partitions the complement of $\mathrm{CH}(P_{k-1})$ into vertical slabs $S(e)$, for each edge $e \in E(P_{k-1})$, and two boundary slabs $S(v_l)$, $S(v_r)$, associated with the respective leftmost and rightmost vertices v_l and v_r of $\mathrm{CH}(P_{k-1})$. The *refined conflict list* of e, C^*_e, is defined as $C^*_e := (P_k \setminus P_{k-1}) \cap S(e)$. We add to this collection the sets $C^*_{v_l} := (P_k \setminus P_{k-1}) \cap S(v_l)$ and $C^*_{v_r} := (P_k \setminus P_{k-1}) \cap S(v_r)$, which we call the refined conflict lists of v_l and v_r, respectively. Note that $C^*_e \subseteq C_e$, for every $e \in E(P_{k-1})$. Moreover, $C^*_{v_l}$ (resp., $C^*_{v_r}$) is contained in $C_{e_1} \cup C_{e_2}$, where $e_1, e_2 \in E(P_{k-1})$ are the two respective edges emanating from v_l (resp., v_r); see Figure 1(a). We now state a key property of the conflict lists C_e (this property is fairly standard and follows from related studies [3]):

Lemma 1 *Let Q be a planar m-point set, r a positive integer satisfying $1 \leq r \leq m$, and $R \subseteq Q$ a random subset of size r. Suppose that $f(\cdot)$ is a monotone non-decreasing function, so that $f(x)/x^c$ is decreasing, for*

[2] The arrangement of a set of planar lines is the decomposition of the plane into *vertices*, *edges* and *faces* (also called *cells*), each being a maximal connected set contained in the intersection of at most two lines and not meeting any other line.

some constant $c > 0$. Then $\mathbf{Exp}\left[\sum_{e\in E(R)} f(c_e)\right] = O\left(r \cdot f\left(\frac{m}{r}\right)\right)$, where c_e is the number of points $p \in Q \setminus R$ in conflict with $e \in E(R)$. □

Constructing the refined conflict lists. We next present how to construct the refined conflict lists at the k-th round of the algorithm. We first construct, in a preprocessing step, the refined conflict lists $C^*_{v_l}$, $C^*_{v_r}$ in overall $O(z_k)$ time. For the sake of the analysis, we eliminate these points from P_k for the time being, and continue processing them only at the final step of the construction—see below.

Let $\mathrm{UH}(P_{k-1})$ be the upper hull of P_{k-1}, and let $\mathrm{LH}(P_{k-1})$ be its lower hull. Having these structures at hand, we construct the zones of $\mathrm{UH}(P_{k-1})$ and $\mathrm{LH}(P_{k-1})$ in $\mathcal{A}(L)$. This takes overall $O(n\alpha(n))$ time, using the vertical decomposition of $\mathcal{A}(L)$ and the fact that the zone complexity of a convex curve in a planar arrangement of n lines is $O(n\alpha(n))$; see [12, Theorem 5.11].

As soon as we have the zones as above, we can determine for each line $\ell \in L$ the edges $e \in E(P_{k-1})$ that ℓ intersects (if any). Let L_1 be the lines that intersect $\mathrm{CH}(P_{k-1})$, and put $L_2 := L \setminus L_1$. (At this stage of the analysis, we ignore all lines corresponding to the points in P_k that were eliminated at the time we processed $C^*_{v_l}$, $C^*_{v_r}$.)

Next, we wish to find, for each point $p \in P_k \setminus P_{k-1}$ the edges in $E(P_{k-1})$ in conflict with p. If p lies inside $\mathrm{CH}(P_{k-1})$, there are no conflicts. Otherwise, we efficiently find an edge $e_p \in E(P_{k-1})$ visible from p, whence we search for the slab $S(e^*_p)$ containing p—see below.

Let us first consider the points on the lines in L_1. Fix a line $\ell \in L_1$, let $p \in P$ be the point on ℓ, and let q_1, q_2 be the intersections between ℓ and the boundary of $\mathrm{CH}(P_{k-1})$. The points q_1, q_2 subdivide ℓ into two rays ρ_1, ρ_2, and the line segment $\overline{q_1q_2}$. By convexity, $\overline{q_1q_2} \subseteq \mathrm{CH}(P_{k-1})$ and the rays ρ_1, ρ_2 lie outside $\mathrm{CH}(P_{k-1})$. Hence, if p lies on $\overline{q_1q_2}$, it must be contained in $\mathrm{CH}(P_k)$. Otherwise, p sees an edge of $E(P_{k-1})$ that meets one of the rays ρ_1, ρ_2, and we thus set e_p to be this edge (which can be determined in constant time); see Figure 1(b).

We next process the lines in L_2. Note that all points on the lines in L_2 conflict with at least one edge in $E(P_{k-1})$, since no line in L_2 meets $\mathrm{CH}(P_{k-1})$. To find these edges we determine for each $\ell \in L_2$ a vertex p_ℓ on the boundary of $\mathrm{CH}(P_{k-1})$ that is extreme for ℓ.[3] This can be done in total time $O(n)$ by ordering $E(P_{k-1})$ and L_2 according to their slopes (the latter being performed during preprocessing), and then merging these two lists in linear time. Next, fix such a line $\ell \in L_2$, and let $p \in \ell$ be a query point, then p must see one of the two edges in $E(P_{k-1})$ incident to p_ℓ (which can be determined in constant time given p_ℓ), and we thus set e_p to be the corresponding edge; see Figure 1(c).

We are now ready to determine, for each point $p \in P_k$ outside $\mathrm{CH}(P_{k-1})$, the slab $S(e^*_p)$ that contains it (note that e^*_p must be vertically visible from p). If e_p is vertically visible from p, we set $e^*_p := e_p$. Otherwise, we walk along (the boundary of) $\mathrm{CH}(P_{k-1})$, starting from e_p and progressing in an appropriate direction (uniquely determined by p and e_p), until the appropriate slab is found. Using cross pointers between the edges and the points, we can thus easily compute C^*_e for each $e \in E(P_{k-1})$. By construction, all traversed edges are in conflict with p, and thus the overall time for this procedure is proportional to the total size of the conflict lists C_e. Recalling that $c_e = |C_e|$, we obtain

$$\mathbf{Exp}\Big[\sum_{e\in E(P_{k-1})} c_e\Big] = O(z_k) = O(n),$$

by Lemma 1 with $f : m \mapsto m$. This concludes the construction of the refined conflict lists.

Computing $\mathrm{CH}(P_k)$. We next describe how to construct the upper hull of P_k, the analysis for the lower hull is analogous. Let $\langle e_1, \ldots, e_s\rangle$ be the edges along the upper hull of P_{k-1}, ordered from left to right. For each e_i, we sort the points in $C^*_{e_i}$ according to their x-order, using, e.g., merge sort, as well as the points in $C^*_{v_l}$, $C^*_{v_r}$. We then concatenate the sorted lists $C^*_{v_l}, C^*_{e_1}, C^*_{e_2}, \ldots, C^*_{e_s}, C^*_{v_r}$, and merge the result with the vertices of the upper hull of P_{k-1}. Call the resulting list Q, and use Graham's scan to find the upper hull of Q in time $O(|Q|)$. This is also the upper hull of P_k. Applying once again Lemma 1 with $f : m \mapsto m \log m$, and putting $c^*_e := |C^*_e|$, $c^*_{v_l} := |C^*_{v_l}|$, $c^*_{v_r} := |C^*_{v_r}|$, and $A > 0$ an absolute constant, the overall expected running time of this step is bounded by

$$\begin{aligned}
&\mathbf{Exp}\Big[A \cdot (c^*_{v_l} \log c^*_{v_l} + c^*_{v_r} \log c^*_{v_r} + \sum_{e\in E(P_{k-1})} c^*_e \log c^*_e)\Big] \\
&\le \mathbf{Exp}\Big[3A \cdot \sum_{e\in E(P_{k-1})} c_e \log c_e\Big] \\
&= O\left(z_k \log\left(z_k/z_{k-1}\right)\right) = O(n),
\end{aligned}$$

because by definition $C^*_e \subseteq C_e$, so $c^*_e \le c_e$, and $c^*_{v_l} \le c_{e_1} + c_{e_2}$ for two edges e_1, e_2 (and similarly for $c^*_{v_r}$). In total, we obtain that the expected time to construct $\mathrm{CH}(P_k)$ given $\mathrm{CH}(P_{k-1})$ is $O(n\alpha(n))$, and since there are $\log^* n$ iterations, the total running time is $O(n\alpha(n)\log^* n)$. We have thus shown:

Theorem 2 *Using $O(n^2)$ space and time, we can preprocess a set L of n lines in the plane, such that given*

[3] By this we mean that p_ℓ is extremal in the direction of the outer normal of the halfplane that is bounded by ℓ and contains $\mathrm{CH}(P_{k-1})$.

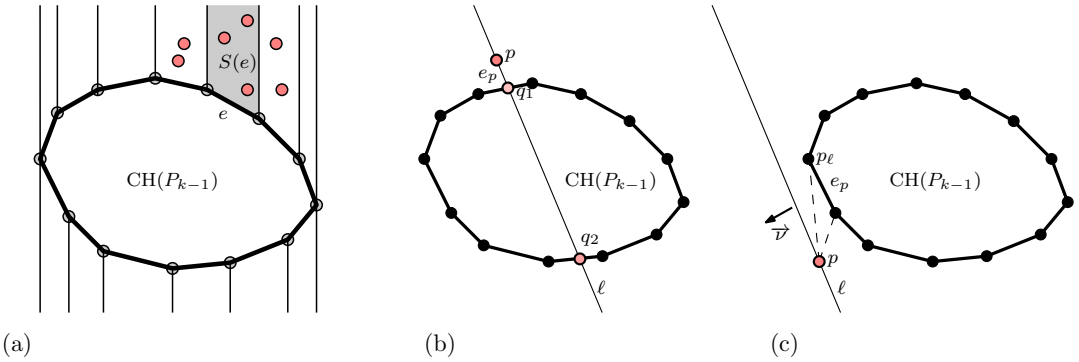


Figure 1: (a) The conflict list C_e of the edge $e \in E(P_{k-1})$ contains all the lightly-shaded points, whereas the refined conflict list $C^*(e)$ has only those points in the vertical slab $S(e)$; (b–c) The edge e_p of $E(P_{k-1})$ is visible to p when (b) ℓ intersects $\mathrm{CH}(P_{k-1})$, or (c) ℓ does not meet $\mathrm{CH}(P_{k-1})$. In this case p_ℓ is an extreme vertex in the direction $\overrightarrow{\nu}$, and the two dashed lines depict the visibility lines between p and the two respective endpoints of e_p.

any point set P with each point lying on a distinct line in L, we can construct $\mathrm{CH}(P)$ in expected time $O(n\alpha(n)\log^* n)$.

We note that the analysis proceeds almost verbatim when L is just a set of *line-segments* in the plane. In this case, we preprocess the lines containing the input segments, and proceed as in the original problem. Omitting the straightforward details, we obtain:

Corollary 3 *If L is a set of n line-segments, we can preprocess L in $O(n^2)$ space and time, such that given a point set P with each point lying on a distinct segment of L, we can construct $\mathrm{CH}(P)$ in expected time $O(n\alpha(n)\log^* n)$.*

Further results. In the full version of the paper, we show that under the "obliviousness assumption" we can improve the expected running time of our algorithm to $O(n\alpha(n))$, while leaving the preprocessing and storage time unchanged. The obliviousness assumption states that the random choices of the preprocessing phase are unknown to the adversary and that the inputs are chosen in a manner oblivious of the current data structure. We also show how to modify the algorithm to make it output sensitive, and how to produce trade-off bounds for storage and running time.

We can extend our result to computing k-sets, and we achieve an improved running time as long as $k = o(\log n)$. Finally, we provide lower bounds in the algebraic computation tree model for related problems, such as preprocessing lines in the plane for sorting a point set according to its x order, where each point in the set lies on a distinct line; and preprocessing a set of planes in $\mathbb{R}^3$ for computing the convex hull of a point set with each point incident to a distinct plane. In all these cases preprocessing does not reduce the $\Theta(n \log n)$ time bound.

Acknowledgments The authors wish to thank Maarten Löffler for suggesting the problem, and Boris Aronov and Timothy Chan for helpful discussions. E. Ezra was supported by a PSC-CUNY Research Award; W. Mulzer was supported in part by NSF grant CCF-0634958, NSF CCF 083279, and a Wallace Memorial Fellowship in Engineering.

References

[1] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational geometry: algorithms and applications*. Springer-Verlag, Berlin, third edition, 2008.

[2] K. Buchin, M. Löffler, P. Morin, and W. Mulzer. Preprocessing imprecise points for Delaunay triangulation: Simplified and extended. *Algorithmica*. To appear.

[3] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry. II. *DCG*, 4(5):387–421, 1989.

[4] O. Devillers. Delaunay triangulation of imprecise points, preprocess and actually get a fast query time. Technical Report 7299, INRIA, 2010.

[5] M. Held and J. S. B. Mitchell. Triangulating input-constrained planar point sets. *IPL*, 109(1):54–56, 2008.

[6] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SICOMP*, 15(1):287–299, 1986.

[7] M. J. van Kreveld, M. Löffler, and J. S. B. Mitchell. Preprocessing imprecise points and splitting triangulations. In *Proc. 19th ISAAC*, pages 544–555, 2008.

[8] M. Löffler. *Data Imprecision in Computational Geometry*. PhD thesis, Utrecht University, 2009.

[9] M. Löffler and W. Mulzer. Triangulating the square and squaring the triangle: quadtrees and Delaunay triangulations are equivalent. In *Proc. 22nd SODA*, pages 1759–1777, 2011.

[10] M. Löffler and J. Snoeyink. Delaunay triangulation of imprecise points in linear time after preprocessing. *CGTA*, 43(3):234–242, 2010.

[11] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *CGTA*, 1(1):51–64, 1991.

[12] M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, New York, NY, USA, 1995.

Learning a 2-Manifold with a Boundary in $\mathbb{R}^3$

Christian Scheffer and Jan Vahrenhold
Faculty of Computer Science, Technische Universität Dortmund, 44227 Dortmund, Germany
{christian.scheffer,jan.vahrenhold}@cs.tu-dortmund.de

Abstract

We present an algorithm for approximating the local feature size of point samples used for reconstructing a 2-manifold in $\mathbb{R}^3$. Our algorithm improves previous results by simultaneously achieving the following two goals: it computes an approximation of the local feature size as well as the number of sample points needed for this approximation using local information only and at the same time is able to deal with 2-manifolds with a smooth boundary.

1 Introduction

Reconstructing a manifold from point samples in $\mathbb{R}^3$ is a fundamental problem that has attracted considerable interest both in Computer Graphics and Computational Geometry. Most algorithms follow the general framework introduced by Amenta and Bern [2] and perform the reconstruction based upon the Voronoi diagram for the sample points. Computing a Voronoi diagram in $\mathbb{R}^3$, however, has a quadratic worst-case complexity, and thus Funke and Ramos [7] present a careful analysis of their variant of the CoCone algorithm to show that a reconstruction is possible in near-linear time. Building upon this, Dumitriu *et al.* [5, 6] propose to first decimate the point set in a preprocessing step using information locally available before computing the (graph) Voronoi diagram for the reduced point set; this algorithm does not rely on CoCone. Recently, Dey *et al.* [4] considered the case of reconstructing a 2-manifold in $\mathbb{R}^3$ that has a smooth boundary.

We demonstrate how to modify the algorithm by Dumitriu *et al.* [5, 6] to be able to reconstruct a 2-manifold in $\mathbb{R}^3$ that has a smooth boundary. As it turns out, we only need to replace one single subroutine, namely the approximation of the so-called *local feature size*, by a variant that can handle (smooth) boundaries. Our algorithm builds upon ideas presented by Funke and Ramos [7] and we derive sufficient conditions for the correctness of the reconstruction and comment on the approximation quality of our algorithm. Unlike the algorithm of Dey *et al.* [4], our algorithm does not rely on a global sampling condition with a sampling constant known to the algorithm but retains the property that the density of the sampling may be locally different.

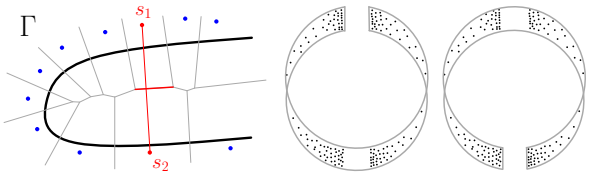
2 Preliminaries

Amenta and Bern [2] introduced the *local feature size* as central concept used for reconstructing smooth (closed) surfaces. For any point x on the manifold Γ, the local feature size $lfs(x)$ is defined as the distance of x to the medial axis of Γ. In different terms, the local feature size in a point $x \in \Gamma$ is the radius of the smaller of the two largest balls touching Γ in x from the inside resp. from the outside and not containing any other point of Γ in their interior. Thus, the local feature size captures the curvature and the folding of Γ. It is known that the (topological) correctness of a reconstruction algorithm depends on the density of the set $S \subset \Gamma$ of sample points used for the reconstruction relative to the local feature size.

Definition 1 *A discrete subset S of a smooth 2-manifold $\Gamma \subset \mathbb{R}^3$ is an ε-sample for Γ iff for every point $x \in \Gamma$ there is a point $s \in S$ with $|xs| \leq \varepsilon \cdot lfs(x)$.*

Dumitriu *et al.*'s algorithm [5, 6] (Algorithm 1) first uses a subroutine from Funke and Ramos [7] to compute an approximation of $\varepsilon \cdot lfs(s)$ for each $s \in S$ and then connects two sample points $s_1, s_2 \in S$ by an edge if and only if $|s_1 s_2| \leq c \cdot \varepsilon \cdot \max\{lfs(s_1), lfs(s_2)\}$ where c is a constant used for compensating for the approximation error. Using the resulting neighborhood graph, the algorithm computes a k-hop-stable subsample, locally identifies the connectivity of the induced subgraph, and computes a topologically correct reconstruction of the manifold. Finally, all sample points not in the subsample are reinserted to refine the reconstruction.

Figure 1(a) shows a situation where the presence of the boundary of Γ (drawn as a thick line) results in an illegal edge $\overline{s_1 s_2}$ to be constructed. Even a higher sampling density close to the boundary does not alleviate this problem–see Figure 1(b).



(a) Illegal edge $\overline{s_1 s_2}$. (b) Non-unique reconstruction.

Figure 1: Problems induced by boundaries.

This is an extended abstract of a presentation given at EuroCG 2011. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

Algorithm 1 Learning a 2-manifold in $\mathbb{R}^3$ [5, 6].

1: Compute an approximation $\phi(s)$ for $\varepsilon \cdot \mathit{lfs}(s)$ for $s \in S$ using a routine by Funke and Ramos [7].
2: Construct a neighborhood graph on S by connecting $s_1, s_2 \in S$ if $|s_1 s_2| \in O(\max\{\phi(s_1), \phi(s_2)\})$.
3: Compute $S_{\text{sub}} \subseteq S$ as a maximal k-hop stable set in the neighborhood graph.
4: Construct the graph Voronoi diagram of S with respect to S_{sub}.
5: Use the graph Voronoi diagram to identify certified adjacencies between points in S_{sub}.
6: Identify faces and triangulate non-triangular faces.
7: Reinsert points from $S \setminus S_{\text{sub}}$ using a routine by Funke and Ramos [7].

One of the key observations by Dumitriu *et al.* [5], however, is that Steps 1 and 2 guarantee that one can reconstruct a manifold from an ε-sample using only information locally available during the remainder of the algorithm (Steps 3–7). The contribution of this work is a replacement for Step 1, i.e. an approximation algorithm for the local feature size that uses only information locally available and can handle the case where the manifold to be reconstructed has a smooth boundary. This, however, requires the sampling condition to be modified.

3 Local feature size for 2-manifolds with boundary

In the remainder of this paper, we will assume that Γ is a smooth 2-manifold in $\mathbb{R}^3$ and that $\partial\Gamma$ is a smooth 1-manifold. We extend the definition of the local feature size to the case of 2-manifolds with a boundary.

Definition 2 *For the medial axis $\mathcal{M}_{\partial\Gamma}$ of the boundary $\partial\Gamma$ and the medial axis $\mathcal{M}_{\Gamma^\circ}$ of the interior Γ° we define*

$$\begin{aligned} \mathit{lfs}_{\partial\Gamma}(x) &= \min\{\operatorname{dist}(x, \mathcal{M}_{\partial\Gamma})\} && \text{for } x \in \partial\Gamma \\ \mathit{lfs}_{\partial\Gamma}(x) &= \min_{y\in\partial\Gamma}\{\mathit{lfs}_{\partial\Gamma}(y) + |xy|\} && \text{for } x \in \Gamma^\circ \\ \mathit{lfs}_{\Gamma^\circ}(x) &= \min\{\operatorname{dist}(x, \mathcal{M}_{\Gamma^\circ})\} && \text{for } x \in \Gamma \end{aligned}$$

The local feature size *of a point $x \in \Gamma$ then is defined as $\mathit{lfs}(x) := \min\{\mathit{lfs}_{\partial\Gamma}(x), \mathit{lfs}_{\Gamma^\circ}(x)\}$.*

Dey *et al.* also define a (similar) variant of the local feature size but then use it to define "a global measure $\rho = \inf_{x\in\Gamma} \mathit{lfs}(x)$" [4, p. 1373] that is know to the algorithm and used for defining the (global) quality of the sample. In contrast, we follow Funke and Ramos [7] and approximate the local feature size locally. The correctness of the algorithms of Funke and Ramos [7] and Dumitriu *et al.* [5, 6] is based upon the observation that the local feature size is a *1-Lipschitz function*, i.e., a non-negative function f with $f(x) \le f(x') + |xx'|$ for all x, x'. Lemma 1 allows us to reuse their (algorithm and) correctness proof for points "far away" from $\partial\Gamma$:

Lemma 1 *The local feature size as defined in Definition 2 is a 1-Lipschitz function.*

Following Funke and Ramos' approach, we consider the Voronoi cell $\mathit{Vor}(s)$ of a sample point s restricted to the (unknown) manifold Γ. One then can show that the distance from s to the furthest vertex v in $\mathit{Vor}(s) \cap \Gamma$ is an approximate lower bound for $\frac{\varepsilon}{1-\varepsilon} \cdot \mathit{lfs}(s)$. Since Γ is unknown to the algorithm, we appoximate $\mathit{Vor}(s) \cap \Gamma$ by $\mathit{Vor}(s) \cap \tilde{T}_s$ where $\tilde{T}_s$ is an approximation of the plane T_s tangent to Γ in s. Since Γ has a boundary, $\partial\Gamma$ can cross $\mathit{Vor}(s)$ such that some Voronoi vertices in $\mathit{Vor}(s) \cap \tilde{T}_s$ do not have a corresponding vertex *on* Γ—see Figure 2.

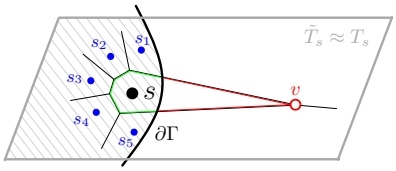


Figure 2: $\partial\Gamma$ crosses $\mathit{Vor}(s)$.

To avoid the lower bound for $\frac{\varepsilon}{1-\varepsilon} \cdot \mathit{lfs}(s)$ to be (arbitrarily) biased by such a cut-off vertex v, the main challenge is to restrict the "search space" for the Voronoi vertices on $\mathit{Vor}(s) \cap \tilde{T}_s$ to areas certified not to contain any part of the boundary or – if it crosses the cell – to exclude cut-off Voronoi vertices while computing the approximation.

4 Safeguarding Voronoi cells from $\partial\Gamma$

To simplify notation, we define what we call a *bicone*; bicones will be shown to contain the locally relevant part of the manifold (including the boundary).

Definition 3 *Let $x_1, x_2 \in \mathbb{R}$, $x_1 \neq x_2$, $r \ge |x_1 x_2|/2$, and let E be a plane containing both x_1 and x_2. Let $C_{E,1}$ and $C_{E,2}$ denote the two circles in E that have radius r and pass through x_1 and x_2. The* bicone *$C^{cone}_{x_1,x_2,r}$ of x_1 and x_2 with parameter r then is defined as the union of the intersections of $C_{E,1}$ and $C_{E,2}$ for all planes E containing x_1 and x_2 (see Figure 3):*

$$C^{cone}_{x_1,x_2,r} := \bigcup_{E \ni x_1, x_2} (C_{E,1} \cap C_{E,2})$$

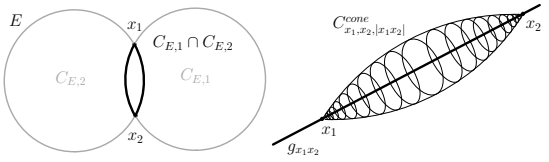


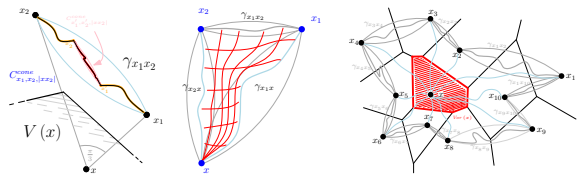
Figure 3: Bicone $C^{cone}_{x_1,x_2,r}$ of x_1 and x_2 w.r.t. r.

In Lemma 5, we name the conditions that guarantee the existence of a smooth curve connecting two points x_1 and x_2 inside a bicone $C^{cone}_{x_1,x_2,r}$ for a properly defined r. For Corollary 2 and Lemmas 3 and 4, we consider the three points $x, x_1, x_2 \in \Gamma$ where $|xx_1|, |xx_2|, |x_1x_2| \leq \frac{1}{2} \cdot lfs(x)$ and where x_1 and x_2 realize an angle of at most $\frac{1}{3} \cdot \pi$ at apex x. Based upon these points, we define $r := \max\{|xx_1|, |xx_2|\}$.

Corollary 2 *There exist three continuous curves $\gamma_{xx_1}, \gamma_{xx_2}, \gamma_{x_1x_2} \subset \Gamma$ connecting x and x_1, x and x_2, and x_1 and x_2 with $\gamma_{xx_1} \subset C^{cone}_{x,x_1,r}$, $\gamma_{xx_2} \subset C^{cone}_{x,x_2,r}$ and $\gamma_{x_1x_2} \subset C^{cone}_{x_1,x_2,r}$.*

Lemma 3 (see Figure 4(a)) *The Voronoi region of x with respect to x_1 and x_2 lies outside of $C^{cone}_{x_1,x_2,r}$.*

Lemma 4 (see Figure 4(b)) *The patch of Γ bounded by $\gamma := \gamma_{xx_1}\gamma_{x_1x_2}\gamma_{x_2x}$ does not contain any point of $\partial\Gamma$ in its interior.*



(a) $\gamma_{x_1x_2}$ does not cross $Vor(x)$. (b) Patch bounded by $\gamma_{xx_1}\gamma_{x_1x_2}\gamma_{x_2x}$. (c) Closed curve γ does not cross $Vor(s)$.

Figure 4: Configurations of Lemmas 3, 4, and 5

Combining the above results, we can show that the Voronoi cell of some (sample) point $x \in \Gamma$ restricted to the intersection with Γ is guaranteed not to contain any point from $\partial\Gamma$ if there are enough (sample) points distributed around it. More precisely, the guarantee holds if every wedge of a ball with radius $\frac{1}{2} \cdot lfs(x)$ centered at x contains at least one other sample point.

Lemma 5 (See Figure 4(c)) *Let $x, x_0, \ldots, x_{\ell-1}$ be points on Γ for which the following holds:*

- $|xx_i| < \frac{1}{2} \cdot lfs(x)$ *for all* $i \in \{0, \ldots, \ell-1\}$.
- $\forall i \in \{0, \ldots, \ell-1\} : \exists j, k \in \{0, \ldots, \ell-1\} \setminus \{i\} :$ $\angle(x_j, x, x_i) \leq \frac{1}{3} \cdot \pi \ \wedge \ \angle(x_k, x, x_i) \geq \frac{5}{3} \cdot \pi$

Then there is no point of $\partial\Gamma$ inside the intersection of Γ and the Voronoi cell of x, i.e.

$$(\Gamma \cap Vor(x))^\circ \cap \partial\Gamma = \emptyset$$

5 Sampling 2-manifolds with a boundary

In Lemma 5, we considered a situation, which allows us to account the (restricted) intersection of $Vor(s)$ with T_s for approximating $\varepsilon \cdot lfs(s)$ by the largest distance of a point $v \in Vor(s) \cap T_s$ to s. This configuration of points describe the distribution of "close" sample points around s. The closeness of these sample points is guaranteed by the following theorem:

Theorem 6 *Let S be an ε-sample of Γ. Then, for each $s \in S$, we have at least $k_{\frac{1}{2},\varepsilon} \in \mathcal{O}(1/\varepsilon^2)$ nearest neighbors $\{s_1, \ldots, s_{k_{\frac{1}{2},\varepsilon}}\} \subset S$ such that $\mathrm{dist}(s, s_i) < \frac{1}{2} \cdot lfs(s)$ for each $s_i \in \{s_1, \ldots, s_{k_{\frac{1}{2},\varepsilon}}\}$.*

The final property of the sample points has to describe a "good" distribution of these $k_{\frac{1}{2},\varepsilon} \in \mathcal{O}(1/\varepsilon^2)$ nearest neighbors around s. Figure 1(b) illustrates that the standard sampling condition does not guarantee a unique reconstruction.

Definition 4 *A discrete subset $S \subset \Gamma$ is an (ε, k)-sample iff S is an ε-sample and the following condition holds for each $s \in S$: for any point $y \in \Gamma$ with $|ys| \in [\frac{\varepsilon}{1+\varepsilon} \cdot lfs(s), \frac{2\cdot\varepsilon}{1-\varepsilon} \cdot lfs(s)]$, there exists a point $s' \in S$ who is one of the k nearest neighbors of s and for which $\angle(y, s, s') \leq \frac{1}{6} \cdot \pi$ hold.*

For an (ε, k)-sample, we can prove that the premise of Lemma 5 holds for every sample point which has at least a distance of $\frac{\varepsilon}{1-\varepsilon} \cdot lfs(s)$ to $\partial\Gamma$. Also, we can compute a restriction H_s of $Vor(s) \cap T_s$ for all other sample points by analyzing the $k \leq k_{\frac{1}{2},\varepsilon}$ nearest neighbors of s, so that $\partial\Gamma$ do not intersect $H_s \cap Vor(s)$.

As mentioned in Section 3, the algorithm relies on an approximation $\tilde{T}_s$ of the plane T_s tangent to a sample point $s \in \Gamma$. Whereas the Funke and Ramos' base algorithm works with every (in an asymptotic sense) good approximation, our variant requires an absolute bound on the deviation of the planes' normals:

Lemma 7 *Let S be an (ε, k)-sample with $\varepsilon \leq \frac{1}{16}$ and $k \leq k_{\frac{1}{2},\varepsilon}$. We can compute an approximation $\tilde{T}_s$ of the plane T_s tangent to Γ in $s \in S$ with $\angle(T_s, \tilde{T}_s) \leq 17°$ by inspecting k nearest neighbours.*

6 Approximating the local feature size

Summarizing the results from Section 4, we obtain the following algorithm for approximating the local feature size for an (ε, k)-sample of a 2-manifold with a boundary (Algorithm 2).

Substituting this algorithm for Step 1 in Algorithm 1 and observing that the remaining steps work on information locally available in the neighborhood graph, we obtain an algorithm for reconstructing a smooth (orientable or non-orientable) 2-manifold in $\mathbb{R}^3$ with a boundary.

Algorithm 2 Approximating $lfs(s)$ for $s \in S$.

Require: S ((ε, k)-sample with $\varepsilon \leq \frac{1}{16}$ and $k \leq k_{\frac{1}{2},\varepsilon}$)

1: **for** each $s \in S$ **do**
2: NH_s :=NN(S,s, k). ▷ Find k NNs of s in S.
3: $\tilde{T}_s$:=TangPlane(NH_s, s). ▷ See Lemma 7.
4: $Vor_{\tilde{T}_s}(s) := Vor(s) \cap \tilde{T}_s$.
5: **if** s does not fulfill Lemma 5's premise **then**
6: $Vor_{\tilde{T}_s}(s) := Vor_{\tilde{T}_s}(s) \cap H_s$.
7: v := Voronoi vertex of $Vor_{\tilde{T}_s}(s)$ furthest to s.
8: $\phi(s) := |vs|$.
9: Return ϕ.

7 Analysis of the algorithm

A direct implementation of Algorithm 2 results in a quadratic running time due to the complexity of computing the Voronoi diagram in $\mathbb{R}^3$. For this version, however, we can prove the following:

Theorem 8 *Algorithm 2 computes $\phi(s)$ for each $s \in S$ such that $\phi(s) \leq 1.135 \cdot \frac{\varepsilon}{1-\varepsilon} \cdot lfs(s)$.*

If one is willing to trade approximation quality for running time, one can (prove and) use that the Voronoi cell of a sample point s with respect to its k nearest neighbors is a "good" approximation for the Voronoi cell used in Algorithm 2 in the sense that the constant in Theorem 8 increases from 1.135 to no more than 1.3. In this case, the dominating step of this algorithm is the computation of the k nearest neighbors, which (using a well-separated pair decomposition [3] and Theorem 6) can be done in $\mathcal{O}(\frac{n}{\varepsilon^2} \log \frac{n}{\varepsilon^2})$ time. With the exception of Step 2 (computing the neighborhood graph), the remaining steps of Algorithm 1 can be executed in the same time complexity, since the subgraph worked with is not too large [6] and re-inserting the excluded points can be done in near-linear time [7]. Step 2 boils down to answering n spherical range queries in $\mathbb{R}^3$ (i.e., n halfspace reporting queries in $\mathbb{R}^4$) which can be done in $\mathcal{O}(n^{4/3} \operatorname{polylog} n)$ time [1, 8].

8 Experimental Evaluation

Since we changed only the first step of Algorithm 1, we were interested in its output, i.e. the approximation of the local feature size. As it is common practice, we first assess the quality by visual inspection: Figure 5 shows a reconstruction of part of the dragon model used in [6] (the graph shows the union of all Delaunay duals for the (restricted) Voronoi cells from Step 7 of Algorithm 2.). We obtained a manifold with a boundary by extracting a slice of the dragon's head.

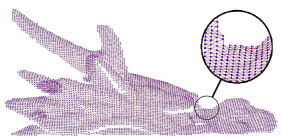


Figure 5: Dragon's head.

In addition to this visual assessment, we also evaluated the approximation quality of our algorithm for estimating the local feature size. To this end, we constructed samples S° and S^S of two manifolds Γ° and Γ^S in $\mathbb{R}^3$ in a way that we were able to derive the local feature size of the manifold from the construction algorithm and thus were able to also numerically assess the quality of our approximation algorithm. By construction, the correct values of ϕ° (for S°) and thus also of ϕ^S (for S^S) can be shown to be approximately 1.85. Our approximation of $\varepsilon \cdot lfs$ computes in both cases values in the range $[1.83, 1.87]$, that is very close to the actual value.

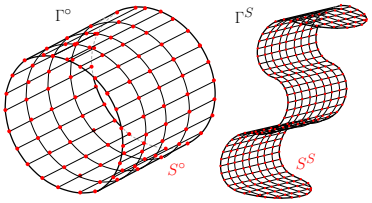


Figure 6: Manifolds and samples.

9 Conclusions

We have presented a new sampling condition that allows for approximating the local feature size of the points on a 2-manifold with a boundary embedded in $\mathbb{R}^3$. The resulting approximation algorithm builds upon previous work but has the advantage of simultaneously being able to handle manifolds with boundaries and not having to assume a global lower bound on the local feature size. The experimental evaluation tightly supports the theoretical analysis of the approximation quality.

References

[1] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In *Advances in Discrete and Computational Geometry*, pages 1–56. AMS, 1999.

[2] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999.

[3] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

[4] T. K. Dey, K. Li, E. A. Ramos, and R. Wenger. Isotopic reconstruction of surfaces with boundaries. *Computer Graphics Forum*, 28(5):1371–1382, 2009.

[5] D. Dumitriu, S. Funke, M. Kutz, and N. Milosavljević. On the locality of extracting a 2-manifold in $\mathbb{R}^3$. In *Proc. Scandinavian Workshop on Algorithm Theory*, LNCS 5124, pages 270–281, Springer, 2008.

[6] D. Dumitriu, S. Funke, M. Kutz, and N. Milosavljević. How much geometry it takes to reconstruct a 2-manifold in $\mathbb{R}^3$. *ACM Journal on Experimental Algorithmics*, 14, May 2009. Article 2.2, 17 pages.

[7] S. Funke and E. A. Ramos. Smooth-surface reconstruction in near-linear time. In *Proc. Symposium on Discrete Algorithms*, pages 781–790. ACM, 2002.

[8] J. Matoušek. Reporting points in halfspaces. *Computational Geometry: Theory and Applications*, 2(3):169–186, 1992.

Approximate Polytope Membership Queries*

Sunil Arya† Guilherme D. da Fonseca‡ David M. Mount§

Abstract

In this paper, we consider an approximate version of a fundamental geometric search problem, polytope membership queries. Given a convex polytope P in $\mathbb{R}^d$, the objective is to preprocess P so that, given a query point q, it is possible to determine efficiently whether q lies inside P subject to an allowed error ε. Previous solutions were based on straightforward applications of classic polytope approximation techniques by Dudley (1974) and Bentley et al. (1982). The former yields minimum storage, and the latter yields constant query time. A space-time tradeoff can be obtained by interpolating between the two. We present a significant improvement to this tradeoff. For example, using the same storage as Dudley, we reduce the query time from $O(1/\varepsilon^{(d-1)/2})$ to $O(1/\varepsilon^{(d-1)/4})$.

To establish the relevance of our results, we introduce a reduction from approximate nearest neighbor searching to approximate polytope membership queries, providing significant improvements to the best known space-time tradeoffs for approximate nearest neighbor searching.

1 Introduction

The problem of determining whether a query point q lies within a convex polytope P, represented as the intersection of halfspaces, is dually equivalent to answering halfspace emptiness queries. Such queries find applications in many geometric problems, such as linear programming queries, ray shooting, nearest neighbor searching, and the computation of convex hulls. In dimension $d \leq 3$, it is possible to build a data structure of linear size that can answer such queries in logarithmic time. In higher dimensions, however, all exact data structures with roughly linear space take $\widetilde{O}(n^{1-1/\lfloor d/2 \rfloor})$ query time, which, except in small dimensions, is little better than brute-force search.

Throughout, we assume that P is a convex polytope lying within the hypercube $[-1,1]^d$, which is represented as the intersection of the set of halfspaces that define its facets. Given $\varepsilon > 0$, we say that P' is an *ε-approximation* to P if the Hausdorff distance between P and P' is at most ε. (Typically, polytope approximation is defined relative to P's diameter, but by scaling P to lie within $[-1,1]^d$, there is no loss of generality in this formulation.) Dudley [6] showed that, for any convex body, it is possible to construct an ε-approximating polytope with $O(1/\varepsilon^{(d-1)/2})$ facets. This bound is asymptotically tight in the worst case.

Given a polytope P, a positive real ε, and a query point q, an *ε-approximate polytope membership query* determines whether q lies inside or outside of P, but it may return either answer if q's distance from P's boundary is at most ε. Approximate polytope membership queries arise in a number of applications, such as collision detection, training a support vector machine, and approximate nearest neighbor searching.

Dudley's construction provides a naïve solution to the approximate polytope membership problem. Construct an ε-approximation P', and determine whether q lies within all its bounding halfspaces. This approach takes $O(1/\varepsilon^{(d-1)/2})$ query time and space. An alternative simple solution was proposed in [2]. Create a d-dimensional grid with cells of diameter ε and, for every column along the x_d-axis, store the two extreme x_d values where the column intersects P. This algorithm produces an approximation P' with $O(1/\varepsilon^{d-1})$ facets. Given a query point q, it is easy to determine if $q \in P'$ in constant time (assuming a model of computation that supports the floor function), but the space required by the approach is $O(1/\varepsilon^{d-1})$.

These two extreme solutions raise the question of whether a tradeoff is possible between space and query time. Before presenting our results, it is illustrative to consider a very simple method for generating such a tradeoff. Given $r \in [\varepsilon, 1]$, subdivide the bounding hypercube into a regular grid of cells of diameter r and, for each cell that intersects the polytope's boundary, apply Dudley's approximation to this portion of the polytope. Subject to minor technical details, the result is a data structure of space $O(1/(\varepsilon r)^{(d-1)/2})$ and query time $O((r/\varepsilon)^{(d-1)/2})$. This interpolates nicely between the two extremes for $\varepsilon \leq r \leq 1$.

*A more complete version will appear in STOC 2011.

†Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, arya@cse.ust.hk.

‡Departamento de Informática Aplicada, Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Brazil, fonseca@uniriotec.br. Research supported by FAPERJ grant E-26/110.091/2010 and CNPq/FAPERJ grant E-26/110.552/2010.

§Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, USA, mount@cs.umd.edu. Research supported by NSF grant CCR-0635099 and ONR grant N00014-08-1-1015.

Given the optimality of Dudley's approximation, it may be tempting to think that the above tradeoff is optimal, but we will demonstrate that it is possible to do better. We show first that it is possible to build a data structure with storage $O(1/\varepsilon^{(d-1)/2})$ (the same as Dudley) that allows polytope membership queries to be answered significantly faster, in $O(1/\varepsilon^{(d-1)/4})$ time. By iterating a suitable generalization of this construction, it is possible to produce a succession of structures of increasingly better search times.

Theorem 1 *Given a polytope P in $[-1,1]^d$, a parameter $\varepsilon > 0$, and a constant $\alpha \geq 2$, it is possible to answer ε-approximate polytope membership queries in time $t = 1/\varepsilon^{(d-1)/\alpha} \geq 1$ with storage space*

$$O\left(1/\varepsilon^{(d-1)\left(1-\frac{1}{2^{\lfloor \log_2 \alpha \rfloor}}-\frac{\lfloor \log_2 \alpha \rfloor - 1}{\alpha}\right)}\right).$$

These space-time tradeoffs are presented visually in Figure 1(a). The lower bound will be discussed later.

The data structure and its construction are both extremely simple. The data structure consists of a quadtree where each leaf cell stores a set of halfspaces whose intersection approximates the polytope's boundary within the cell. A query is answered by performing a point location in the quadtree followed by a brute force inspection of the halfspaces in the node. The data structure is constructed by the following recursive algorithm, called SplitReduce. It is given the polytope P, the approximation parameter ε, and the desired query time t. The initial quadtree box is $Q = [-1,1]^d$.

SplitReduce(Q):

1. Let P' be an ε-approximation of $Q \cap P$.
2. If the number of facets $|P'| \leq t$, then Q stores the hyperplanes bounding P'.
3. Otherwise, split Q into 2^d quadtree boxes and invoke SplitReduce on each such box.

Although the algorithm itself is deceptively simple, the analysis of the storage as a function of the query time t is nontrivial. The storage efficiency depends on the assumption that the number of facets of P' is approximately minimal. This will be made precise in Section 2. In addition to proving upper bounds on the space complexity of the above algorithm, we will establish a lower bound (see Figure 1(a) and Theorem 6).

To establish the relevance of our results, we introduce a reduction from approximate nearest neighbor searching to approximate polytope membership queries, providing significant improvements to the best known space-time tradeoffs for approximate nearest neighbor searching [1]. Our reduction implies the following improved space-time tradeoffs for ANN queries.

Theorem 2 *Given set of n points in $\mathbb{R}^d$, a parameter $\varepsilon > 0$, and a constant $\alpha \geq 2$. There is a data structure for approximate nearest neighbor searching with query time $O(\log n + (\log(1/\varepsilon))/\varepsilon^{d/\alpha})$ and storage*

$$O\left(n/\varepsilon^{d\left(1-\frac{1}{2^{\lfloor \log_2 \alpha \rfloor}}-\frac{\lfloor \log_2 \alpha \rfloor}{\alpha}\right)}\right).$$

These space-time tradeoffs together with known upper bounds and lower bounds [1] are illustrated in Figure 1(b). Although the connection between the polytope membership problem and ANN has been noted before by Clarkson [5], we are the first to provide a reduction that holds for point sets with unbounded aspect ratio. Details are omitted in this version.

2 Upper Bound for Polytope Membership

In this section, we present upper bounds for the storage of the data structure obtained by the SplitReduce algorithm for a given query time t. We may assume that P is presented succinctly to our algorithm, for example, as the output of Chan's coreset construction [3]. We start by discussing the first step of the algorithm.

Step 1 consists of obtaining a polytope P' that ε-approximates $Q \cap P$. Some polytopes can be approximated with far fewer than $\Theta(1/\varepsilon^{(d-1)/2})$ facets. The problem of approximating a polytope with the minimum number of facets reduces to a set cover problem [7] and an $O(\log(1/\varepsilon))$ approximation can be obtained by a greedy algorithm. Clarkson [4] showed that if c is the smallest number of facets required to approximate P, then we can obtain an approximation with $O(c \log c)$ facets in $O(nc^2 \log n \log c)$ randomized time, where n is the number of facets of P.

For simplicity, we assume that the algorithm used in Step 1 produces an approximation to $Q \cap P$ with a minimum number of facets. (In the full version it will be shown that an application of Dudley's algorithm to an appropriate subset of P suffices for our purposes. An alternative is to run Clarkson's approximation algorithm on $Q \cap P$. This increases the query time by a negligible factor of $O(\log(1/\varepsilon))$.)

For completeness, we now describe Dudley's algorithm. Let P be a polytope, and let the *size* σ_P of P denote the side length of the smallest axis aligned box Q that contains P. We assume that the center of Q is the origin. Dudley's algorithm obtains an approximation polytope P' in the following manner. Let B be a ball of radius $\sigma_P\sqrt{d}$ centered at the origin. Place a set J of $\Theta((\sigma_P/\varepsilon)^{(d-1)/2})$ points on the surface of B such that every point on the surface of B is within distance $O(\sqrt{\varepsilon\sigma_P})$ of some point in J. For each point $j \in J$, determine the nearest facet of P and add the corresponding halfspace to P'. We consider that each point $j \in J$ generates a facet of P' of diameter at most

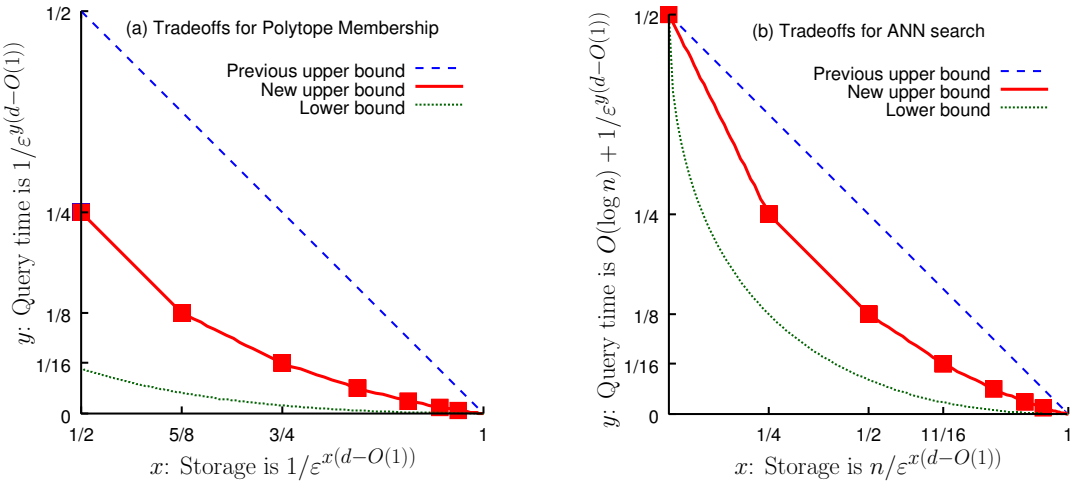


Figure 1: The multiplicative factor in the exponent of the $1/\varepsilon$ term for (a) polytope membership queries and (b) approximate nearest neighbor (ANN) queries. The $O(1)$ term in the exponent corresponds to a constant that does *not* depend on d.

$\sqrt{\varepsilon\sigma_P}$, even if these facets are coplanar. The following lemma follows from standard results on Dudley's algorithm.

Lemma 3 *Given a polytope P of size σ_P, Dudley's algorithm produces a polytope P' with $O((\sigma_P/\varepsilon)^{(d-1)/2})$ facets that approximates P. Furthermore, all facets of P' have diameter at most $\sqrt{\varepsilon\sigma_P}$.*

Recall that our algorithm takes four inputs: polytope P, box Q, query time t, and approximation error ε. For simplicity, we refer to the algorithm as SplitReduce(Q), since the parameters P, t, ε remain unchanged throughout the recursive calls. The output of our algorithm is a quadtree whose leaf cells induce a subdivision of Q. Each leaf cell L stores an approximation of $P \cap L$ with $t_L \leq t$ facets, where t_L is the minimum number of facets required to approximate $P \cap L$. The storage of this quadtree is defined as the total number of stored facets over all the leaf cells. Before we prove the whole space-time tradeoff, we show that the algorithm produces a data structure with query time $t = \Theta(1/\varepsilon^{(d-1)/4})$ and the same storage as Dudley's algorithm, $O(1/\varepsilon^{(d-1)/2})$.

Lemma 4 *The output of SplitReduce(Q) for $t \geq (\sigma_Q/\varepsilon)^{(d-1)/4} \geq 1$ is a quadtree with storage $O((\sigma_Q/\varepsilon)^{(d-1)/2})$.*

Proof. Let T denote the quadtree produced by the algorithm. For each leaf cell L of T, let t_L be the number of facets stored in L. We will show that $\sum_L t_L \leq (\sigma_Q/\varepsilon)^{(d-1)/2}$, which establishes the storage bound and proves the lemma.

Towards this end, we first prove a lower bound on the size of any leaf cell L. We assert that there exists a constant c_1 such that every leaf cell L has size $\sigma_L \geq \sqrt{\varepsilon\sigma_Q/c_1}$. The assertion follows from Lemma 3. In particular, the standard Dudley technique applied to a cell of size $\sqrt{\varepsilon\sigma_Q/c_1}$ produces a polytope with at most $c_D(\sigma_Q/c_1\varepsilon)^{(d-1)/4}$ facets, where c_D is the constant arising from Dudley's method. By choosing c_1 to be a sufficiently large constant, the number of facets is at most t and the termination condition of our algorithm implies that such a cell is not subdivided.

Let P_D be the polytope obtained by applying Dudley's algorithm to $P \cap Q$. Combining our assertion with Lemma 3, each facet of P_D intersects $O(1)$ leaf cells. We assign each facet to all leaf cells that it intersects. The correctness of Dudley's algorithm implies that the facets assigned to any cell L provides an approximation of $P \cap L$. Thus, t_L is no more than the number of Dudley facets assigned to L. Since the number of facets of P_D is $O((\sigma_Q/\varepsilon)^{(d-1)/2})$, it follows that $\sum_L t_L = O((\sigma_Q/\varepsilon)^{(d-1)/2})$, which completes the proof. □

We now use the previous lemma as a base case in order to extend the space-time tradeoff to other query times.

Theorem 5 *Let $\alpha \geq 2$ be a constant. The output of SplitReduce(Q) for $t = (\sigma_Q/\varepsilon)^{(d-1)/\alpha} \geq 1$ is a quadtree with storage*

$$O\left(\left(\frac{\sigma_Q}{\varepsilon}\right)^{(d-1)\left(1-\frac{1}{2^{\lfloor \log_2 \alpha \rfloor}}-\frac{\lfloor \log_2 \alpha \rfloor - 1}{\alpha}\right)}\right).$$

Proof. Let $k = \lfloor \log_2 \alpha \rfloor$. Our proof proceeds by induction on a constant number of steps k. The base case case $k = 1$ corresponds to Lemma 4. Next, assume that the theorem holds for $1, \ldots, k-1$, that is, for $\alpha < 2^k$. We need to prove that the theorem holds for $2^k \leq \alpha < 2^{k+1}$.

Let T denote the quadtree produced by the algorithm with $2^k \leq \alpha < 2^{k+1}$. Let T' denote the subtree induced by cells of size at least $\sqrt{\varepsilon\sigma_Q}/2$. Arguing exactly as in the proof of Lemma 4, the sum $\sum_L t_L$ over all leaf cells of T' is $O((\sigma_Q/\varepsilon)^{(d-1)/2})$. The leaf cells of T' that have $t_L \leq t$ are not refined by the algorithm and their total storage clearly satisfies the bounds of the theorem.

We now consider the subset $\mathcal{L}$ of leaf cells of T' such that for $L \in \mathcal{L}$ we have $t_L > t$. By Markov's inequality, $|\mathcal{L}| = O((\sigma_Q/\varepsilon)^{(d-1)/2})/t = O((\sigma_Q/\varepsilon)^{(d-1)\left(\frac{1}{2}-\frac{1}{\alpha}\right)})$. Also, the size of the cells in $\mathcal{L}$ is between $\sqrt{\varepsilon\sigma_Q}/2$ and $\sqrt{\varepsilon\sigma_Q}$, since larger cells would have been subdivided.

Recall that the induction hypothesis states that the theorem holds for $\alpha < 2^k$. Since the size of the cells L in $\mathcal{L}$ is $\sigma_L \leq \sqrt{\varepsilon\sigma_Q}$, we have $t = (\sigma_L/\varepsilon)^{(d-1)/\alpha'}$, where $\alpha' \leq \alpha/2$ for sufficiently small $\varepsilon \leq \sigma_Q/4$ (if $\varepsilon > \sigma_Q/4$, then the theorem holds trivially). Therefore we can use the induction hypothesis to obtain the following storage for each cell $L \in \mathcal{L}$:

$$O\left(\left(\frac{\sigma_Q}{\varepsilon}\right)^{(d-1)\left(\frac{1}{2}-\frac{1}{2^k}-\frac{k-2}{\alpha}\right)}\right).$$

We then multiply by $|\mathcal{L}| = O((\sigma_Q/\varepsilon)^{(d-1)\left(\frac{1}{2}-\frac{1}{\alpha}\right)})$ completing the induction. □

Note that the height of the quadtree obtained by SplitReduce is $O(\log(1/\varepsilon))$, since cells of size ε are never subdivided. Therefore, it is straightforward to locate the quadtree cell containing the query point in $O(\log(1/\varepsilon))$ time. Theorem 1 follows as a straightforward consequence.

3 Lower Bound for Polytope Membership

In this section, we present lower bounds on the space-time tradeoffs obtained by our algorithm for polytope membership. Our main result is the following.

Theorem 6 *Let $\alpha \geq 2$ be a constant. There exists a polytope P such that the output of SplitReduce$([-1,1]^d)$ for $t = 1/\varepsilon^{(d-1)/\alpha} \geq 1$ is a quadtree with storage*

$$\Omega\left(1/\varepsilon^{(d-1)\left(1-2\sqrt{\frac{2}{\alpha}}+\frac{3}{\alpha}\right)-1}\right).$$

Our approach is similar to the lower bound proof of [1]. It is based on constructing a hypercylinder that takes dimension k, $1 \leq k \leq d-2$ as a parameter, and bounding the storage requirements of our data structure for it. We carry out this analysis in Lemma 7.

Lemma 7 *Let k and t be integers, where $1 \leq k \leq d-2$, $1 \leq t \leq 1/\varepsilon^{(d-1)/2}$, and let $0 < \varepsilon \leq 1$. There exists a polytope P such that the output of SplitReduce$([-1,1]^d)$ for P is a quadtree with storage $\Omega\left(t\left(\frac{1}{\varepsilon t^{2/(d-k-1)}}\right)^k\right)$.*

Proof. (sketch) Consider an infinite polyhedral hypercylinder C whose "axis" is a k-dimensional linear subspace K defined by k of the coordinate axes, and whose "cross-section" is a polytope P_0 that approximates a $(d-k)$-dimensional ball of diameter Δ. If we choose $\Delta = O(\varepsilon t^{2/(d-k-1)})$, then any polytope P_0' that approximates P_0 requires at least $(2^d+1)t$ facets. Define polytope P to be the truncated cylinder obtained by intersecting C with the hypercube $[-1,1]^d$.

Consider a set $\mathbb{X}$ of points that are at least Δ apart placed on the intersection of $[-1,1]^d$ with the k-dimensional axis of the hypercylinder C. By a simple packing argument, we can ensure that the number of points in $\mathbb{X}$ is at least $\Omega(1/\Delta^k)$. Let $P_0^{\mathbb{X}}$ denote the set of cross-sections of C passing through the points of $\mathbb{X}$. Consider the set of leaf cells of the quadtree that overlap any cross-section $P_0 \in P_0^{\mathbb{X}}$. Recall from our construction that these cells must together contain at least $(2^d+1)t$ facets. We say that a leaf cell is *small* if its size is at most Δ. By a packing argument, at least t facets are contained in the small leaf cells intersecting P_0. Noting that small leaf cells cannot intersect two cross-sections of $P_0^{\mathbb{X}}$, since they are at least Δ apart, it follows that the total space used by all the small leaf cells together is at least $\Omega(t|\mathbb{X}|) = \Omega(t/\Delta^k) = \Omega(t(1/(\varepsilon t^{2/(d-k-1)}))^k)$, which proves the lemma. □

Setting k appropriately, we obtain the lower bound in Theorem 6.

References

[1] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM*, 57:1–54, 2009.

[2] J. L. Bentley, F. P. Preparata, and M. G. Faust. Approximation algorithms for convex hulls. *Commun. ACM*, 25(1):64–68, 1982.

[3] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom.*, 35(1):20–35, 2006.

[4] K. L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3rd Workshop Algorithms Data Struct. (WADS)*, pages 246–252, 1993.

[5] K. L. Clarkson. An algorithm for approximate closest-point queries. In *Proc. 10th ACM Symp. Comput. Geom. (SoCG)*, pages 160–164, 1994.

[6] R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Approx. Theory*, 10(3):227–236, 1974.

[7] J. S. B. Mitchell and S. Suri. Separation and approximation of polyhedral objects. *Comput. Geom.*, 5:95–114, 1995.

Certified Computation of Morse-Smale Complexes on Implicit Surfaces

A. Chattopadhyay * G. Vegter†

Abstract

The Morse-Smale complex is an important tool for global topological analysis in various problems in computational topology and data analysis. A certified algorithm for computing the Morse-Smale complexes has been presented for two-dimensional Morse-Smale systems in bounded planar domains [3]. In the current article we extend the approach in case of Morse-Smale systems on two-dimensional manifolds (surfaces) which are given by regular level sets of smooth functions. We give an outline of our method for computing certified Morse-Smale complexes on the implicit surfaces and present some implementation results.

1 Introduction

Extracting global topological information of an implicit surface is an important problem in computational topology. One way to extract the global topological information of an implicit surface is by approximating the surface with a simplicial complex followed by a computation of its topological invariants, such as homology groups, Betti numbers, genus etc. There are many existing computational topology tools [4] for doing such an analysis. A second and more direct approach is to analyze the gradient field of a height function on an implicit surface. However, analyzing the gradient field of the height function can be done via computing its Morse-Smale (MS) complex, which is a configuration of singular points and separatrices of the corresponding gradient system. The MS-complex of a gradient system reveals the global topology of the underlying shape. Since it is known that almost all height functions are Morse functions, i.e., functions having only non-degenerate critical points, and by applying a small perturbation, a Morse function could be transformed into a MS function (defined in 2). Therefore, computation of the topologically correct MS-complex of a height function, or more generally, of a MS-function on an implicit surface is a worthwhile problem.

Existing algorithms for MS-complexes can compute the complex of a piecewise linear manifold or, in other words, of a discrete gradient-like vector field [6]. The topological correctness depends on how coarse the discretization is. On the other hand, we consider the problem of computing a *certified* approximation of the MS-complex of a smooth implicit MS-function. In other words we compute a configuration of piecewise-linear curves which is isotopic to the MS-complex on the implicit surface. Our algorithm uses interval arithmetic to bring the computation into the context of Exact Geometric Computation (EGC) paradigm [8].

Our contribution. The main difference of the current article with the previous article [3] is that in the current problem the MS-complex is known only implicitly on an implicit surface. In particular, the algorithm for computing the certified approximation of MS-complexes determines the followings.

- isolated *certified boxes* each containing a unique saddle, source or sink, and each box has the local topological property of a saddle, source or sink; we give a novel approach for computing certified boxes corresponding to critical points of a MS-function by considering intersections of a underlying Jacobi set with the implicit surface.
- *certified initial and terminal intervals* (corresponding to a saddle), each of which is guaranteed to contain a unique point corresponding to a saddle-source or saddle-sink connector (separatrix);
- disjoint *certified strips*, lying on the implicit surface, around each separatrix, each of which containing exactly one separatrix and can be made as close to the separatrix as desired.

2 Preliminaries

Interval arithmetic (IA). Interval arithmetic is used to cope with rounding errors in finite precision computations. A *range function* $\square F$ for a function $F : \mathbb{R}^m \to \mathbb{R}^n$ computes for each m-dimensional input interval I (i.e., an m-box) an n-dimensional output interval $\square F(I)$, such that $F(I) \subset \square F(I)$. A range function is said to be *convergent* if the diameter of the output interval converges to 0 when the diameter of the input interval shrinks to 0.

Morse function. A function $h : \mathcal{M} \subset \mathbb{R}^3 \to \mathbb{R}$, defined on an implicit manifold $\mathcal{M}$, is called a *Morse function* if all its critical points are non-degenerate. The *Morse lemma* [6] states that near a non-degenerate critical point a it is possible to choose

*Department of Mathematics and Computing Science, University of Groningen, A.Chattopadhyay@rug.nl

†Department of Mathematics and Computing Science Science, University of Groningen, G.Vegter@rug.nl

local co-ordinates x, y in which h is expressed as $h(x, y) = h(a) \pm x^2 \pm y^2$. The number of minus signs is called the index $i_h(a)$ of h at a. Thus a two variable Morse function has three types of non-degenerate critical points: minima (index 0), saddles (index 1) and maxima (index 2).

Integral line. An integral line $x : I \subset \mathbb{R} \to \mathcal{M}$ passing through a point p_0 on $\mathcal{M}(\subset \mathbb{R}^3)$ is the unique maximal curve satisfying: $\dot{x}(t) = \mathbf{grad}\, h(x(t))$, $x(0) = p_0$, for all $t \in I$. Here the gradient is defined with respect to the metric inherited from $\mathbb{R}^3$. Integral lines corresponding to the gradient vector field of a smooth function $h : \mathcal{M} \to \mathbb{R}$ have many interesting properties, such as: (1) any two integral lines are either disjoint or coincide; (2) an integral line $x : I \to \mathcal{M}$ through a point p of h is injective and if $\lim_{t \to \pm\infty} x(t)$ exists, it is a critical point of h; (3) the function h is strictly increasing along the integral line of a regular point of h and integral; (4) regular integral lines are perpendicular to regular level sets of h.

Stable and unstable manifolds. Consider the integral line $x(t)$ passing through a point p. If the limit $\lim_{t \to \infty} x(t)$ exists, it is called the ω-limit of p and is denoted by $\omega(p)$. Similarly, $\lim_{t \to -\infty} x(t)$ is called the α-limit of p and is denoted by $\alpha(p)$ – again provided this limit exists. The stable manifold of a singular point p is the set $W^s(p) = \{q \in \mathcal{M} \mid \omega(q) = p\}$. Similarly, the unstable manifold of a singular point p is the set $W^u(p) = \{q \in \mathcal{M} \mid \alpha(q) = p\}$ [7]. Here we note that both $W^s(p)$ and $W^u(p)$ contain the singular point p itself [7]. The components of $W^s(p) \setminus \{p\}$ ($W^u(p) \setminus \{p\}$) are called stable (unstable) separatrices. Each saddle has two stable and two unstable separatrices.

The Morse-Smale (MS) complex. A Morse function on $\mathcal{M}$ is called a MS-function if its stable and unstable manifolds intersect *transversally*. Since $\mathcal{M}$ is a 2-dimensional surface, this means that stable and unstable separatrices are disjoint. In particular, there are no saddle-saddle connections. The MS-complex associated with a MS-function h on $\mathcal{M}$ is the subdivision of $\mathcal{M}$ formed by the connected components of the intersections $W^s(p) \cap W^u(q)$, where p, q range over all singular points of h. According to the quadrangle lemma [6], each region of the MS-complex is a quadrangle with vertices of index $0, 1, 2, 1$, in this order on the boundary of the region.

Jacobi set. Let us consider two Morse functions $f, H : \mathbb{R}^3 \to \mathbb{R}$. Let c be a regular value of f and then by implicit function theorem $\mathcal{M}_c := f^{-1}(c)$ is a smooth 2-manifold. Then generically, the restriction h_c of H to the regular level set $\mathcal{M}_c$ of f is a Morse function [5]. The critical points of H restricted to a level set of the function f correspond to points where ∇f is a multiple of ∇H, i.e., the points where the Jacobian of the map $\mathcal{F} = (f, H) : \mathbb{R}^3 \to \mathbb{R}^2$ has rank < 2. The Jacobi set $\mathbb{J}(f, H)$ (or $\mathbb{J}$ for short) is the closure of the set of critical points of such level set restrictions $\mathbb{J}(f, H) = cl\{\mathbf{x} \in \mathcal{M}_c : \mathbf{x}$ is a critical point of $h_c\}$, for any regular value $c \in \mathbb{R}$. From the Smooth Embedding Theorem [5] we have, generically, that the Jacobi set of two Morse functions $f, H : \mathbb{R}^3 \to \mathbb{R}$ is a smoothly embedded 1-manifold in $\mathbb{R}^3$.

3 Method

Problem set-up. Let $f : \mathbb{R}^3 \to \mathbb{R}$ be a $\mathcal{C}^2$ function, and let zero be a regular value of f. Then the set $\mathcal{M} = \{\mathbf{x} \equiv (x, y, z) \in \mathbb{R}^3 : f(\mathbf{x}) = 0\}$ is a regular surface (Follows from the *implicit function theorem*). Let $H : \mathbb{R}^3 \to \mathbb{R}$ be a Morse function such that H has no critical point on $\mathcal{M}$. Then, generically, the function $h : \mathcal{M} \to R$, defined by $h = H|_{\mathcal{M}}$, is a Morse function. We assume h is a MS-function. Then the corresponding gradient field is a MS-system. We are interested in computing the MS-complex of $\mathbf{grad}\, h$.

Gradients on surfaces. First we analyze the vector field $\mathbf{grad}\, h$ by finding its singularities and their types. We have assumptions: (1) zero is a regular value of f and (2) H has no critical point on $\mathcal{M} = f^{-1}(0)$. For a $\mathcal{C}^2$ function h on a compact manifold $\mathcal{M}$ the gradient vector field $\mathbf{grad}\, h$ on $\mathcal{M}$ is characterized as follows: for $p \in \mathcal{M}$ and $\mathbf{grad}\, h(p) \in T_p\mathcal{M}$, $\langle \mathbf{grad}\, h(p), v \rangle = d_v h(p), \quad \forall v \in T_p\mathcal{M}$. Here, $d_v h$ is called the directional derivative of h along v. Again from figure 1, we note that

$$\begin{aligned} \mathbf{grad}\, h(p) &= proj_{\mathcal{M}} \nabla H(p) = \nabla H(p) - (\nabla H(p).\hat{n})\hat{n} \\ &= \nabla H(p) - \frac{\nabla f(p) \cdot \nabla H(p)}{\|\nabla f(p)\|^2} \nabla f(p). \end{aligned}$$

Here $\hat{n}$ is the unit normal vector at point p of the implicit surface $\mathcal{M}$.

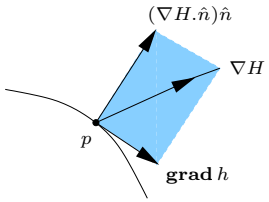


Figure 1: Tangential and normal components of ∇H.

Local expression in a parametric domain. Let $f_z(p) \neq 0$ at $p \in \mathcal{M}$. Then locally near p the manifold $\mathcal{M}$ can be parametrized by a map $\tau : U \to \mathbb{R}^3$ of the form $\tau(x, y) = (x, y, g(x, y))$, where U is an open subset of $\mathbb{R}^2$. Let $q \in U$ be the point in U corresponding to p, i.e., $\tau(q) = p$. Let us consider $\tilde{h}(x, y) = H(x, y, g(x, y))$, locally in U. Then we denote the local gradient field as $\nabla\tilde{h}$ where $\tilde{h}_x = H_x - \frac{f_x}{f_z} H_z$ and

$\tilde{h}_y = H_y - \frac{f_y}{f_z} H_z$. Similarly, one can derive local expressions of the gradient field on the yz and zx-plane, by assuming $f_x \neq 0$ and $f_y \neq 0$ respectively.

Again the integral curves of $\mathbf{grad}\, h$ (on $\mathcal{M}$) are projected onto integral curves of the vector field $\mathrm{grad}_{\tau^* G} \tilde{h}$ on $U \subseteq \mathbb{R}^2$ given by,

$$X = \frac{1}{f_x^2 + f_y^2 + f_z^2} \begin{pmatrix} (f_y^2 + f_z^2)\, H_x - f_x f_y\, H_y - f_x f_z\, H_z \\ -f_x f_y\, H_x + (f_y^2 + f_z^2)\, H_y - f_y f_z\, H_z \end{pmatrix}.$$

In particular, the integral curves of $\mathbf{grad}\, h$ are mapped onto those of $\mathrm{grad}_{\tau^* G} \tilde{h}$ by the projection τ^{-1} of the neighborhood $\tau(U)$ of p in $\mathcal{M}$ onto the xy-plane.

Note that a singular point $(x_0, y_0, z_0) \in \mathcal{M}$ of $\mathbf{grad}\, h$ has the same type (saddle, sink or source) as the point (x_0, y_0), considered as a singular point of $\nabla \tilde{h}$ (even though the integral curves of the latter gradient are different from the projections of the integral curves of $\mathbf{grad}\, h$). We use this observation in the following analysis of saddles, sinks and sources.

To determine the type (minima, maxima or saddle) and nature (degenerate or non-degenerate) of the critical points we consider the bordered Hessian matrix, which is expressed as:

$$\mathcal{H}L = \begin{pmatrix} 0 & -f_x & -f_y & -f_z \\ -f_x & L^{xx} & L^{xy} & L^{xz} \\ -f_y & L^{xy} & L^{yy} & L^{yz} \\ -f_z & L^{xz} & L^{yz} & L^{zz} \end{pmatrix}.$$

where $L^{xx} = H_{xx} - \frac{H_z}{f_z} f_{xx}$ etc. Let $\det \mathcal{H}L$ be the corresponding determinant. The following theorem [9] gives a necessary and sufficient condition for h to have a critical point.

Theorem 1 *Let X be open in $\mathbb{R}^3$ and let $f, H : X \to \mathbb{R}$ be functions of class C^2 such that 0 is a regular value of f. Let $\mathcal{M}$ be the manifold $\{\mathbf{x} \in X : f(\mathbf{x}) = 0\}$ and let $\mathbf{x}_0$ be a point on $\mathcal{M}$. Then h (the restriction of H to $\mathcal{M}$) has a critical point at $\mathbf{x}_0$ iff there exists some scalar λ such that $\nabla H(\mathbf{x}_0) = \lambda \nabla f(\mathbf{x}_0)$.*

Monotonicity of f on $\mathbb{J}$. Finally, for certified isolation of a critical point of h inside a box I we first prove that along each component of the Jacobi set $\mathbb{J}$, inside I, function f is monotonic in the following theorem:

Theorem 2 *Consider a box I, in the domain of the function f, satisfying (i) $I \cap \mathcal{M} \neq \emptyset$, (ii) $\nabla f \neq 0$, (iii) $\nabla H \neq 0$, (iv) $\det \mathcal{H}L \neq 0$. Then if $I \cap \mathbb{J} \neq \emptyset$, 1. $\mathbb{J}$ is regular, 2. $\mathbb{J}$ can have at most finitely many components inside I, 3. Each component of $\mathbb{J}$ is transversal to $\mathcal{M}$ inside I, 4. f is monotonic along each component of $I \cap \mathbb{J}$.*

Isolating and detecting critical points of h. Here we describe the first step of our MS-complex computation method. Let us consider a three-dimensional bounding box, say B, that contains the implicit surface $\mathcal{M}$. Now the subdivision algorithm subdivides the box B until for each subinterval, I say, it is possible to determine whether h has a unique critical point inside I or not. The subdivision process is driven by the following set of conditions:

(i) $C_1 : 0 \in \Box f(I)$

(ii) $C_2 : 0 \in \Box \nabla H(I) \times \Box \nabla f(I)$

(iii) $C_3 : \langle \Box \nabla f(I), \Box \nabla f(I) \rangle > 0$

(iv) $C_4 : 0 \notin \Box \nabla H(I)$

(v) $C_5 : 0 \notin \det \Box \mathcal{H}L(I)$

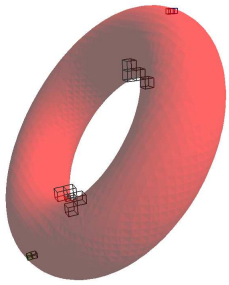


Figure 2: Finding Critical Boxes by domain subdivision

In the subdivision process, if $\neg C_1$ holds for an interval I then I does not contain a zero of f and we stop further subdivision of I. In the subdivision process, if $\neg C_2$ holds, then we stop further subdividing that interval I, since by Theorem 1, in that case I is not a possible candidate for containing a critical point of h. C_3 is a *small normal variaton* condition ensuring parametrizability of $\mathcal{M} \cap I$. C_4 ensures the non-singularity of function H in I. Finally, C_5 ensures that the number of critical points of h in I is at most one. The following subdivision algorithm isolates the critical points of a MS-function h in a bounding box B

Algorithm. SearchCritical(h, B)

1. Initialize an octree $\mathcal{T}$ to the bounding box B.
2. Subdivide $\mathcal{T}$ until for all the leaves I we have:
$$\neg C_1 \vee \neg C_2 \vee (C_3 \wedge C_4 \wedge C_5).$$
3. For each leaf I_l
4. Do if C_1, C_2, C_3, C_4 and C_5 hold then
5. Check if $\mathbb{J}(f, H) \cap \mathcal{M} \neq \emptyset$ inside I_l.
6. If intersection found, the leaf contains a critical point of h. Denote this leaf as I_c.
7. Else, the leaf does not contain a critical point of h.

Now the following corollary of theorem 2 gives a strategy to determine whether inside I_l $\mathbb{J}(f, H) \cap \mathcal{M}$ is $\emptyset$ or not.

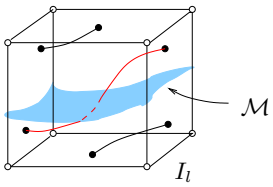


Figure 3: Components of the Jacobi set inside I_l.

Corollary 3 *Let I_l be a leaf interval, obtained by* SearchCritical *(I_l can contain at most one critical point of h). Assume that (i) the Jacobi set $\mathbb{J}(f,H)$ intersects the faces of I_l transversally, and (ii) there is no critical point of h lying on a face of I_l. Let n_{pos} be the number of intersection points of $\mathbb{J}(f,H)$ with the six faces of I_l, where f is positive. Again let n_{neg} be the number of intersection points of $\mathbb{J}(f,H)$ with six faces of I_l, where f is negative. If both n_{pos} and n_{neg} are odd numbers, then there is exactly one critical point of h inside I_l. Otherwise, there is no critical point of h inside I_l.*

Here we note that the assumptions in the corollary 3 ensure only the non-degenerate cases. In the full article, cf [2], we describe how to handle degenerate cases with small perturbation of interval I.

Refinement of intervals containing critical points. Next we further refine the boxes I_c as in the planar vector field situation, see [3, 2]. In the current situation we use the vector field X projected onto one of the faces of I_c, and after refinement use the corresponding pull-back vector field on $\mathcal{M}$.

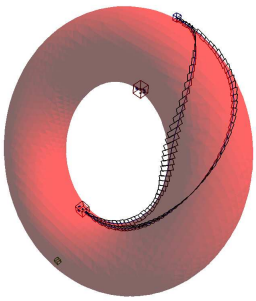
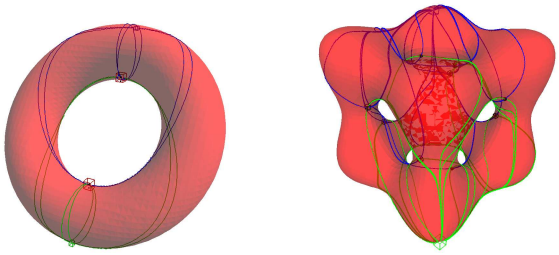


Figure 4: Representing a curve on the implicit surface by the intersection of plane-segments and the surface.

Algorithm for computing separatrices. We compute small rectangular plane-segments, transversal to the implicit surface, such that the intersections of these pieces of planes and the implicit surface together represent a boundary curve of the strip (Figure 4) which contains a separatrix. The gradient vector field **grad** h must satisfy an orientation property which is along the left (for the right boundary curve) and along the right (for the left boundary curve). This will ensure that each separatrix lies in the corresponding strip. The details of this method are discussed in the full version of the article, cf [2].

Implementation results. We implemented of our algorithm using `Boost` library [1] for IA. All experiments have been performed on a 3GHz Intel Pentium 4 machine under Linux with 1 GB RAM using the `g++` compiler, version 3.3.5. Figure 5 show MS-complexes consisting of saddles (red boxes), minima (green boxes) and maxima (blue boxes). Moreover, a pair of blue boundary curves contains a certified unstable separatrix (saddle-maximum connection) and a pair of green boundary curves contains a stable separatrix (saddle-minimum connection).



(a) (b)

Figure 5: Certified MS-complexes on implicit functions given by $f_1(x,y,z) \equiv (x^2+y^2+z^2-R^2-r^2)^2-4R^2(r^2-x^2)=0$ and $f_2(x,y,z) \equiv x^4-5x^2+y^4-5y^2+z^4-5z^2+10=0$ where we consider $H(x,y,z)=z$.

Conclusion. We give a novel approach for computing the certified MS-complex on implicit surfaces. In the end some implementation results are given to give a proof of concept and to justify those algorithms.

References

[1] `Boost` interval arithmetic library. `http://www.boost.org`.

[2] A. Chattopadhyay. *Certified Geometric Computation: Radial Basis Function based Isosurfaces and Morse-Smale Complexes.* PhD thesis, University of Groningen, 2010. ISBN: 978-90-367-4757-8.

[3] A. Chattopadhyay, S.J. Holtman, and G. Vegter. Certified computation of planar Morse-Smale complexes. In *Abstracts of the 26th European Workshop on Computational Geometry*, pages 105–108, Dortmund, Germany, March 22-24 2010.

[4] H. Edelsbrunner and J. Harer. *Computational Topology. An Introduction.* Amer. Math. Soc., Providence, Rhode Island, 2010.

[5] H. Edelsbrunner and J. Harer. Jacobi Sets of Multiple Morse Functions. In *In Foundations of Computational Mathematics*, pages 37–57, Minneapolis 2002, eds.

[6] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete Comput. Geom*, 30:87–107, 2003.

[7] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, 1974.

[8] C. Li, S. Pion, and C. Yap. Recent progress in Exact Geometric Computation. *J. of Logic and Algebraic Programming*, 64(1):85–111, 2004. Special issue on "Practical Development of Exact Real Number Computation".

[9] S. Plantinga and G. Vegter. Computing contour generators of evolving implicit surfaces. *ACM Transactions on Graphics.*, 25:1243–1280., 2006.

Author Index

Supported by