

On Counting Lines rather than Pages

Michael Hoffmann

Department of Computer Science, ETH Zürich, Zürich, Switzerland
hoffmann@inf.ethz.ch

Irina Kostitsyna

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
i.kostitsyna@tue.nl

1 Abstract

To make the time-constrained review process of scientific conferences feasible, the length of paper submissions—or rather, of the part of submissions to be considered by all reviewers—must be bounded. Such a bound in turn is based on a criterion to measure the length of a paper. Traditionally, almost always the number of pages is used as a measure because it is very easy to determine, and also due to practical and financial implications for preparing printed proceedings.

As the days of physically printed proceedings are over, only ease of use remains as a benefit of the pagecount measure, along with tradition. This benefit should be weighed against several undesirable consequences of exclusively focusing on the number of pages: this measure is not robust under changes of the document style, it encourages authors to cram and overload their pages, and it greatly punishes the use of illustrations, tables, and displaystyle formulae. Therefore, we want to discuss and explore alternative measures for paper length to address some of these shortcomings, without sacrificing ease of use.

Starting from SoCG 2019 we will use the *number of lines* in the text as a measure. While this number cannot be as easily determined as the number of pages, it is quite straightforward to obtain a consistent line numbering using the `lineno` package, which is used and enabled by default in the `lipics-v2019` L^AT_EX-class. A consistent line numbering has the additional benefit that it is easy for reviewers to point to specific parts of the paper for feedback and discussion.

In this note, we discuss some ramifications and the fine print of such a line number measure. We also give some technical hints so as to hopefully make it easy for authors to number and count the lines in their submissions consistently.

Acknowledgements We thank Gill Barequet, Mark de Berg, Erin Chambers, Joseph S.B. Mitchell, Bettina Speckmann, Monique Teillaud, and Yusu Wang for their support. The listed authors are the current maintainers of this document and the accompanying class file.

1 How Do We Count?

Let us start by discussing in a bit more detail which lines are to be counted. The short answer is: Every single line, starting from the abstract header line and up to the line just before “References” (just as here in this note).

Most of these lines should be considered, numbered, and counted correctly by `lineno` [1]. But—depending on the environments, packages and macros used—there may also be certain parts of papers that `lineno` does not handle correctly automatically. In this context, authors should think of `lineno` not as a judge that certifies the number of lines in the paper, but as a tool that helps them to get the numbering and the overall count right. So it is the author’s responsibility to make a decent effort and possibly adjust their L^AT_EX-code so that the lines in these parts are numbered and counted correctly as well—or at least so that the `lineno` count yields a very good approximation.

In order to minimize the effort required, we provide a wrapper class `socg-lipics-v2019` around the `lipics-v2019` document class [4] that hopefully addresses most of the commonly encountered issues with line numbering. In Section 2 we give a short advice how authors should get started. Then in Section 3 we discuss in more detail what exactly `socg-lipics-v2019`



© Michael Hoffmann and Irina Kostitsyna;
 licensed under Creative Commons License CC-BY
 November 17, 2020.



Leibniz International Proceedings in Informatics
 Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

38 does and—to some extent—how it works. Section 4 describes how to add line numbering to
 39 a nested `minipage` environment, which may be helpful as a blueprint if someone wants to
 40 extend `lineno` numbering to some custom environment. In Section 5 we list a few issues
 41 that authors may run into and what can be done about it (or not). Finally, in Section 6 we
 42 discuss our reasoning for the change to count lines rather than pages and summarize the
 43 results from our discussions.

44 Now for a slightly more precise answer to the initial question: What counts?

45 **Frontmatter and Bibliography.** Neither the frontmatter with title and author data nor the
 46 bibliography counts. In this way, papers with many authors do not suffer anymore from the
 47 blown-up frontmatter dimensions in the latest LIPICs document class.

48 **Figures.** By default `lineno` does not number and count certain lines. For instance, figures
 49 are not counted and neither are captions. Not counting figures is intentional because they
 50 do not contain text (other than maybe labels, coordinates, or similar) but they consist of
 51 graphical elements. Also, usually figures contain supplemental information only, in form
 52 of examples, overviews, diagrams, constructions, etc. that could also be removed from the
 53 paper without crippling its contents. But all captions should be numbered and counted.

54 **Tables, Footnotes, etc.** Similar to figures, `lineno` does not handle tables and footnotes
 55 by default. But they should be counted, just as everything else. As an exception, tables do
 56 **not** count if they contain data only (usually, numbers). But they should be counted if they
 57 contain text. If in doubt whether or not the content is data, it is to be counted as text.

58 **Final version.** The line limit also applies to the final version of the paper. Therefore, also
 59 the final version must be submitted with proper line numbering so that the proceedings chair
 60 can easily verify compliance. The published paper, however, will be without line numbers.
 61 So the proceedings chair will disable line numbering there. The `socg-lipics-v2019` class
 62 provides an option `nolineno` that disables line numbering and basically runs the plain LIPICs
 63 class only. Using this option, the authors can see and check how their final published paper
 64 will look like.

65 **2 What should Authors Do?**

66 In short, there are three things:

- 67 (1) Get the [lipics-v2019 authors package](#) from LIPICs [4] and the [socg-lipics-v2019](#)
 68 [class file](#) from the [Computational Geometry Pages](#). Note that the latest LIPICs class
 69 (`lipics-v2019` from December 10, 2018) is needed, earlier versions do not work.
 70 If you want to start from a template, use the sample article file from the LIPICs authors
 71 package, but replace the document class `lipics-v2019` by `socg-lipics-v2019`.
- 72 (2) Use the `socg-lipics-v2019` wrapper class around the standard `lipics-v2019` document
 73 class as your main document class. It attempts to provide a more consistent line
 74 numbering by fixing issues with various command and environments. The next section
 75 goes in some detail over the different issues addressed by this wrapper. Some features
 76 can be switched off separately, in case they give trouble.
- 77 (3) Do **not** use `$$... $$` to typeset displaymath formulae! Use `\[... \]` instead! You will
 78 find various arguments for this advice on the Internet (see, for instance [l2tabu \[8\]](#)), but
 79 our main reason here is that the plain-TEX primitive `$$` does not work well with `lineno`.

80 **3** What Does the `socg-lipics-v2019` Class Do?

81 In this section we list the different tweaks that the `socg-lipics-v2019` wrapper class applies
 82 to the standard `lipics-v2019` document class. This is intended mostly as a documentation
 83 for those who are interested to know exactly what happens. It should also help people to
 84 figure out what goes wrong in case of problems, to handle their custom macros in a similar
 85 fashion, or to suggest improvements. In order to use this class, you do not necessarily need
 86 to read through this section, you can simply use, for instance,

```
87 \documentclass[a4paper,USenglish,cleveref,autoref,thm-restate]{socg-lipics-v2019}
```

88 and see what happens.

89 **Frontmatter and Bibliography.** To disregard the frontmatter, `socg-lipics-v2019` disables
 90 line numbering there by replacing the `\maketitle` command. Also the lines that contain
 91 subject classifications, etc. are considered part of the frontmatter and, therefore, not
 92 numbered. The DOI entry is disabled. An entry *Lines* showing the number of lines in the
 93 paper is shown instead. It is computed from the `linenumber` where the bibliography starts.
 94 As for the bibliography, it does not hurt to leave the numbers there so that reviewers can
 95 refer to them. Just remember that these lines do not count toward the 500 lines bound.

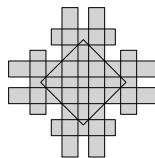
96 **Captions.** The `lineno` package provides a command `\internallinenumbers` to enable line
 97 numbering in internal vertical mode, such as in a float. Using commands from the `caption`
 98 package [5] (that is required by the `LIPICs` class), we hook a call to `\internallinenumbers`
 99 into every caption text.

100 In a similar fashion, this command can be used to extend line numbering to some other
 101 environments that `lineno` does not handle by default; see the example for `minipage` in
 102 Section 4. Note that `\internallinenumbers` assumes a fixed height of lines. So it does not
 103 work well in connection with `displaystyle math`, for example. Within the scope of captions
 104 that should not be an issue, though.

105 Due to the way \TeX handles floats, numbering them is tricky because their position in
 106 the input may differ from their position in the output. The line numbers assigned by `lineno`
 107 correspond to the spot where the figure appears in the input. For instance, Figure 1 appears
 108 in the input file right below this paragraph, and that is how its lines are numbered.

111 At least the map : output line \rightarrow line number is injective and the overall count works
 112 out. Unless \LaTeX places the figure in the middle of a paragraph, the line numbering can be
 113 made consecutive by moving the figure in the input to the position where it appears in the
 114 output. (This is nice to have but not required.)

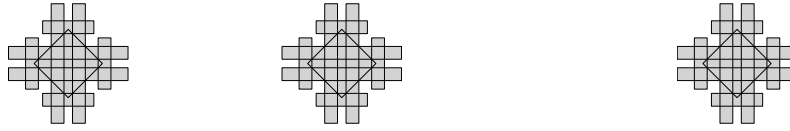
115 **Subcaptions.** The `caption` package offers a `\subcaption` command to combine several
 116 sub-figures into one single figure. While such an aggregation should not be necessary anymore



109 **Figure 1** This figure is placed at the bottom of the page. But the caption line numbers correspond
 110 to the place where it appears in the input \LaTeX -file.

4 On Counting Lines rather than Pages

117 just to save page-space, it is still useful as a means to structure a collection of related figures.
 118 By default, `socg-lipics-v2019` numbers all subcaptions, using the same line number(s) for
 119 subcaptions that appear along the same output line. Also the `subfigure` environment is
 120 handled accordingly. Figure 2 below shows an example. In case this feature gives trouble, it
 121 can be switched off using the documentclass option `nosubfigcap`.



122 (a) a short caption 122 (b) 122 (c) This subfigure has a slightly longer caption that
 123 spans several lines.

124 ■ **Figure 2** How to use and number a figure that consists of several subfigures with subcaptions.

125 For reference, this is how Figure 2 appers in the \LaTeX sourcecode.

```

126 \begin{figure}[thbp]
127   \begin{minipage}[t]{.25\linewidth}
128     \centering\includegraphics[scale=0.4]{socg-logo}
129     \subcaption{a short caption}\label{fig:2:1}
130   \end{minipage}
131   \begin{minipage}[t]{.25\linewidth}
132     \centering\includegraphics[scale=0.4]{socg-logo}
133     \subcaption{}\label{fig:2:2}
134   \end{minipage}
135   \begin{minipage}[t]{.48\linewidth}
136     \centering\includegraphics[scale=0.4]{socg-logo}
137     \subcaption{This subfigure has a slightly longer caption that spans
138       several lines.}\label{fig:2:3}
139   \end{minipage}
140   \caption{How to use and number a figure that consists of several subfigures
141     with subcaptions.}\label{fig:2}}
142 \end{figure}

```

143 **Tables.** The fix described above for figures addresses all captions. For the actual table
 144 contents, the `edtable` package is convenient. To get proper line numbers for a standard table
 145 environment such as `tabular`, it can be wrapped into into an `edtable`. For instance, the code
 146 in Table 1 (left) is effectively processed as shown in Table 1 (right) to appear in the output
 147 as shown in Table 2. By default, `socg-lipics-v2019` wraps all `tabular` environments into
 148 an `edtable`. To globally disable this wrapping, use the documentclass option `notab`.

<pre> 149 \begin{tabular}{ c c c }\hline 150 1 & happy & line \\ \hline 151 2 & happy & line \\ \hline 152 \end{tabular} </pre>	<pre> \begin{edtable}{tabular}{ c c c }\hline 1 & happy & line \\ \hline 2 & happy & line \\ \hline \end{edtable} </pre>
---	--

153 ■ **Table 1** \LaTeX -code for a table using `tabular` in the original version (left) and wrapped into an
 154 `edtable` (right).

155

1	happy	line
2	happy	line

156

157 ■ **Table 2** The table from Table 1 in the output, properly numbered by `lineno`.

160 **Footnotes.** Similar to floats, footnotes are tricky because their placement is determined at
 161 the end of a page only, and it usually differs from their position in the input. By default,
 162 `lineno` does not number them. In order to fix this, `socg-lipics-v2019` wraps the contents
 163 of every footnote into a `minipage`, which is then numbered using `\internallinenumbers`.¹

166 Similar to captions, the numbering with respect to footnotes is not consecutive along the
 167 page. While for captions this often can be fixed by moving the corresponding figure, table,
 168 or caption in the source file, this does not really work for footnotes.² But at least we obtain
 169 a unique line number and a correct overall count.

170 **Arrays.** By default, arrays and its relatives are numbered as a single line. This is fine in
 171 many cases, for instance, where a matrix appears as a single entity, or if there are a few lines
 172 only that are sparsely filled compared to a full line of text. The definition in Line 175 below
 173 is such an example.

$$174 \quad f(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ is even;} \\ 3n + 1 & \text{if } n \text{ is odd.} \end{cases}$$

175 But in other cases, the amount and/or density of content in such a structure may not
 176 be appropriately accounted for by a single line of text. In such a case, the authors should
 177 number the lines. There are different ways to achieve this. For instance, the `amsmath`
 178 environments `align`, `flalign`, `gather` and `alignat`, along with their starred variants, are
 179 properly numbered by `lineno`; see the example in lines 183–185 below. (The example is
 180 short and sparse still, as the text would easily fit into a single line. It is intended to illustrate
 181 the numbering only, not a desperate need for it due to the excessive amount of content.)

$$182 \quad y = x + 2$$

$$183 \quad z \geq x - 1$$

$$184 \quad f(x, y, z) = x + y + z$$

185 Just like `tabular`, an `array` is not numbered by default. It can be wrapped into an `edtable`,
 186 though the mathmode command has to be embedded. For instance, the \LaTeX -code

```
187 \begin{edtable}[$$]{array}{rcl}
188   y & = & x+2 \\
189   z & \geq & x-1 \\
190 \end{edtable}
```

158 ¹ This is an insightful footnote. Of course, it needs a proper line number. The number is “stolen” from
 159 the beginning of the paragraph where the footnote is referenced, not inserted at the end of the page.

164 ² There is a package `fnlineno` that supports numbering footnotes properly. Alas, it only works for
 165 pagewise numbering and it does not seem to cooperate well with `\internallinenumbers`.

6 On Counting Lines rather than Pages

191 generates the following output.

$$\begin{array}{l} 192 \qquad y = x + 2 \\ 193 \qquad z \geq x - 1 \end{array}$$

194 However, this wrapping does not work from within mathmode, which makes it a bit clumsy
195 to use. Therefore, `socg-lipics-v2019` does not perform any automatic wrapping of `arrays`.

196 **Algorithms.** Both the `algorithms` package [2] and the `algorithmicx` package [7] pro-
197 vide two environments `algorithmic` and `algorithm` to format pseudocode. The class
198 `socg-lipics-v2019` adds line numbers to captions and to the `algorithmic` environment
199 using `\internallinenumbers`. See Algorithm 1 below for an example using the `algorithms`
200 package. This feature is enabled only if the packages `algorithm` and `algorithmic(x)`,
201 respectively, are loaded in the preamble of the document. It can be disabled with the
202 documentclass option `noalgorithms`.

203 ■ **Algorithm 1** Example using the `algorithms` package.

```
204 Require:  $n \geq 0$   
205 Ensure:  $n = 0$   
206   while  $N \neq 0$  do  
207     if  $N$  is even then  
208        $N \leftarrow N/2$   
209     else  $\{N$  is odd $\}$   
210        $N \leftarrow N - 1$   
211     end if  
212   end while
```

213 **Algorithm2e.** The `algorithm2e` package [3] provides an environment to format pseudocode.
214 It has its own version of many standard macros, such as line numbers and captions. Its
215 customization options do not seem powerful enough to achieve a style that is consistent with
216 both `LIPICs` and `lineno`. Therefore, `socg-lipics-v2019` changes some internal macros of
217 `algorithm2e` so as to (1) obtain `linenumbers` for both the code (using `algorithm2e`'s own
218 numbering option) and the caption (using `lineno`) and (2) change the appearance to fit
219 with `LIPICs` and `lineno`. Algorithm 2 below illustrates a resulting layout. This feature is
220 enabled only if the package `algorithm2e` is loaded in the preamble of the document. It can
221 be disabled with the documentclass option `noalgorithm2e`.

222 ■ **Algorithm 2** Example pseudocode using `algorithm2e`.

```
223 Data: some input  
224 Result: some output  
225 while not done do  
226   | compute some stuff;  
227   | if something happens then  
228   | | do this;  
229   | else  
230   | | do something else;  
231   | end  
232 end
```

233 **Tcolorbox.** The `tcolorbox` package [6] provides an environment for colored and framed text
 234 boxes. The `socg-lipics-v2019` class handles these environments by adding the command
 235 `\internallinenumbers` and adjusting the spacing to avoid overlap between line numbers and
 236 the surrounding box. This feature is enabled only if the package `tcolorbox` is loaded in the
 237 preamble of the document. It can be disabled with the documentclass option `notcolorbox`.

238 This text is wrapped into `\begin{tcolorbox} ... \end{tcolorbox}`. Such a simple
 239 example is handled fine by `socg-lipics-v2019`. If you work with more elaborate
 240 custom boxes, you may have to do some manual tuning yourself.

241 4 How to (Maybe) Handle Custom Environments

242 The `socg-lipics-v2019` class attempts to handle a number of frequently occurring issues
 243 with `lineno`. But, depending on what packages and macros people use, they may run into
 244 issues that are not covered there. In such a case, it makes sense to check whether there is an
 245 easy workaround with some minor amount of manual tweaking. As a typical example let us
 246 consider the `minipage` environment because (1) it can be easily adopted to get some line
 247 numbers going and (2) it can be used as a tool to handle other issues, by wrapping contents
 248 into a `minipage`. In essence, this is what most of the fixes in `socg-lipics-v2019` do.

249 So let us consider a `minipage` with some regular text inside as an example. By default
 250 `lineno` numbers it is a single line.

251 The text in this box is put into a `minipage`, surrounded by an `fbox`, without
`\internallinenumbers`. It is numbered as a single line containing the
 (multiline) `fbox`. This is technically correct, but not semantically.

252 This is not quite what we want. The text should be considered as three lines. So let us
 253 add the command `\internallinenumbers` inside the `minipage`, which yields the following
 254 result.

255 The text in this box is put into a `minipage`, surrounded by an `fbox`. Using
 256 `\internallinenumbers`, we obtain a proper numbering. But the outer
 257 line is still numbered, resulting in a double numbering.

259 This looks somewhat better, as the inner box is correctly numbered using three lines.
 260 But the outer label for the whole box remains, which does not make sense. Hence we
 261 temporarily switch off line numbering on the outer level by wrapping the whole construct
 262 into a `nolinenumbers` environment. As a result, we obtain the intended line numbering.

263 The text in this box is put into a `minipage`, surrounded by an `fbox`,
 264 using `\internallinenumbers`, and wrapped into `\begin{nolinenumbers}`
 265 `... \end{nolinenumbers}` to avoid double numbering.

266 For reference, here is the \LaTeX -sourcecode for this last version.

```
267 \begin{nolinenumbers}
268   \fbox{
269     \begin{minipage}{.9\textwidth}\internallinenumbers
270       The text in this box is put into a \texttt{minipage}, surrounded by an
```

```

271     \texttt{fbox}, using \verb|\internallinenumbers|, and wrapped into
272     \verb|\begin{nolinelnumbers}| \dots \verb|\end{nolinelnumbers}| to avoid
273     double numbering.
274     \end{minipage}
275   }
276 \end{nolinelnumbers}

```

277 5 Specific Questions & Issues

278 In this section we discuss a few very specific issues that authors may encounter and—if an
 279 easy resolution is known—how to address them.

280 5.1 L^AT_EX Error: File ‘lipics-v2019.cls’ not found

281 As described in Section 2, you need both, the `lipics-v2019` authors package and the
 282 `socg-lipics-v2019` class file. The error message indicates that you have not installed the
 283 files from the `lipics-v2019` authors package into the right location.

284 5.2 Lines Entry Does not Always Update

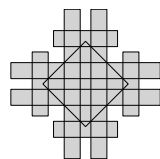
285 Similarly to citations and references, you need to run L^AT_EX twice to see the correct value in
 286 the “Lines” entry. In the first L^AT_EX run the current number of lines is written into the `.aux`
 287 file, and the second run updates the document with the correct value.

288 5.3 Zero Lines

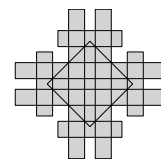
289 If even after multiple runs of L^AT_EX, the “Lines” entry on the titlepage remains zero, then it
 290 is most likely because you do not have a bibliography. Putting a bibliography command and
 291 running `bibtex` should fix this problem. The reason is that the “Lines” entry is computed
 292 from the start of the bibliography because everything following from that point on should
 293 not count anymore. As a result, the entry is meaningful only if there is a bibliography.

294 5.4 Figures Side by Side

295 Consider the example below, where Figure 3 and 4 are placed side by side. The lines in both
 296 captions are numbered, effectively counting these lines twice.



297 ■ **Figure 3** This caption spans several lines 299
 298 that are numbered correctly.



299 ■ **Figure 4** The lines in this caption are also
 300 numbered, leading to an overcount.

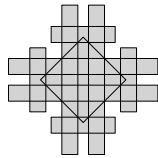
301 In order to avoid this, we would like to number the lines of the longest of these captions
 302 only. Fortunately, it is easy to switch off line numbering for a single caption. The class
 303 `socg-lipics-v2019` implements line numbering using a customization option of the `caption`
 304 package [5]. More precisely, it defines a `textformat` called `socgnumberitall` and sets this
 305 to be the default. So, placing the command

306 `\captionsetup{textformat=simple}`

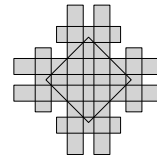
307 right before a `\caption` command switches back to the default, nonnumbered layout, as
 308 shown for Figure 5 and 6.

309 Another, possibly better option is to combine these figures into one single figure and use
 310 `\subcaption` to label (and number) them; see the corresponding paragraph in Section 3.

311 Below is the corresponding L^AT_EX-sourcecode.



312 ■ **Figure 5** This caption spans several lines
 313 that are numbered correctly. These line num-
 314 bers are implicitly shared with Figure 6.



312 ■ **Figure 6** The lines in this caption are not
 313 numbered, implicitly reusing the line numbers
 314 from Figure 5.

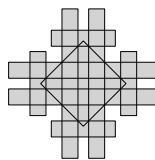
```

315 \begin{figure} [htbp]
316 \begin{minipage}[t]{.45\textwidth}
317   \centering
318   \includegraphics[scale=.5]{socg-logo}
319   \caption{This caption spans several lines that are numbered correctly. These
320     line numbers are implicitly shared with \figurename~\ref{fig:6}.\label{fig:5}}
321 \end{minipage}
322 \hfill
323 \begin{minipage}[t]{.48\textwidth}
324   \centering
325   \includegraphics[scale=.5]{socg-logo}
326   \captionsetup{textformat=simple}
327   \caption{The lines in this caption are not numbered, implicitly reusing the
328     line numbers from \figurename~\ref{fig:5}.\label{fig:6}}
329 \end{minipage}
330 \end{figure}

```

331 5.5 The Last Line of a Paragraph is not Numbered

334 Consider, for instance, the current paragraph. Its last line appears to be unnumbered. As a
 335 compensation there is a spurious line number right after Figure 7.



332 ■ **Figure 7** A figure may disturb the line numbering for the previous paragraph if it is not properly
 333 separated from that paragraph.

335 To avoid such effects, always separate floats from the surrounding text by properly ending
 336 and starting the corresponding paragraphs, for instance, by leaving an empty line in between.
 337 This was not done for the paragraph above, as its source code shown below reveals.
 338

```

339 [...]
340 there is a spurious line number right after \figurename-\ref{fig:7}.
341 \begin{figure}[htbp]
342 [...]

```

343 Adding an empty line before `\begin{figure}` properly ends the preceding paragraph
344 and fixes the problem.

345 5.6 Weird Line Number Placements

346 In some situations `lineno` may seem to place the line numbers at weird spots. Consider,
347 for instance, the following text that appears within a `minipage` and is numbered using
348 `\internallinenumbers`, as discussed in Section 4.

349	The text in this box ... uses <code>\internallinenumbers</code> .
350	$\sum_{i=1}^n i^2 = \dots$
351	
352	
353	There are too many numbers and they are not placed correctly.

354 The reason is, as mentioned earlier, that `\internallinenumbers` assumes a fixed height
355 of lines and, therefore, does not work all that well for this text, which contains a `displaymath`
356 formula. Unfortunately, there does not seem to be an easy workaround. But if this concerns
357 only a few lines of text, then you can add the line numbers manually, by putting the command
358 `\socgnl` (where the last two letters stand for “number line”, not for a country code) at the
359 beginning of each line. A longer part of height uniform text could also be wrapped into a
360 nested `minipage` and numbered using `\internallinenumbers`, of course. Fixing the above
361 box along these lines leads to the following code. (The macro `\cprotect` is only needed
362 because of the internal use of `\verb`.)

```

363 \begin{nolinelnumbers}
364   \begin{center}
365     \noindent\cprotect\fbbox{%
366       \noindent\begin{minipage}{.9\hsize}
367         \socgnl The text in this box is numbered manually using \verb|\socgnl|.
368         \[
369           ~\socgnl\sum_{i=1}^ni^2 =\ldots
370         \]
371         \begin{minipage}{\hsize}\internallinenumbers
372           This paragraph consists of a longer text that is typeset using lines of
373           fixed height. It is wrapped into a nested minipage and collectively
374           numbered using \verb|\internallinenumbers|.
375         \end{minipage}
376       \end{minipage}
377     }
378   \end{center}
379 \end{nolinelnumbers}

```

380 The resulting layout is given below.

381 The text in this box is numbered manually using `\soggnl`.

$$382 \sum_{i=1}^n i^2 = \dots$$

383 This paragraph consists of a longer text that is typeset using lines of fixed
 384 height. It is wrapped into a nested minipage and collectively numbered using
 385 `\internallinenumbers`.

386 The horizontal placement of the line numbers can be adjusted by changing the variable
 387 `\linenumbersep`.

388 **6 Why?**

389 A brief summary of our reasoning has already been given in the abstract. Here is a more
 390 detailed version with some additional bits of information, for the interested reader and as a
 391 base for future discussions. So, if you have thoughts on the matter, please let us know!

392 **Motivation.** Let us start with the motivation to change the current measure. Pagecount
 393 encourages authors to maximize the density of information per page. It encourages the use
 394 of text walls with little or no space in between, and it discourages the use of `displaystyle`
 395 `math`, proper sectioning and paragraph macros, and figures.

396 Specifically figures come at a very high cost. As a consequence, often they are left
 397 out entirely or downscaled and condensed, with detrimental consequences for aesthetics,
 398 clarity, and ultimately usefulness. In particular, papers on nonclassical topics, which need
 399 to introduce more background to be somewhat accessible to nonspecialists, and papers
 400 that propose new directions and models suffer because they rely on illustrative examples
 401 to motivate and explain their concepts and choices. Well designed figures and examples
 402 are an integral part of any geometrically inspired exposition, and as readers—reviewers or
 403 otherwise—we usually wish to have more rather than less of them. But our figure-punishing
 404 pagecount measure forces authors in the diametrically opposite direction.

405 In a similar fashion, this reasoning applies to the other items mentioned: As readers, we
 406 prefer a proper paragraph spacing and `displaystyle` formulae, even if it means that the paper
 407 is four pages longer as a result. To us, pagecount is a measure from an age where all papers
 408 where printed on actual paper. Of course, such printing still happens and should continue to
 409 be possible. But most copies are read electronically nowadays. Therefore, it is due time to
 410 at least consider alternative measures.

411 The overarching goal is to measure the amount of content in a way that is independent
 412 from the typographical layout. This separation between content and typography is a main
 413 strength of a system like \LaTeX .

414 **Alternative Measures.** Linecount is an obvious candidate, which has the advantage of
 415 being fairly easily implementable. Using the `lineno` package to provide line numbers is the
 416 default in LIPICs, anyway, and line numbers are independently desirable to make it easier to
 417 refer to specific parts of the paper in reviews and discussions.

418 Wordcount is the obvious competitor. It is a standard measure in many other areas, such
 419 as humanities and professional publishing. Many tools are available, but it seems hard to
 420 get any two of them to agree on a count for a given document. Specifically, two typical
 421 shortcomings of these tools we found to be blockers:

- 422 ■ They mostly fold on \LaTeX -macros. While most tools recognize macros to some extent,
423 this recognition usually results in discarding these macros from consideration. This
424 makes sense in general, given that many macros do not translate to a word in the output.
425 However, some macros eventually do result in words being added to the output, and
426 simply disregarding those is an error. In particular, any user-defined custom macro is
427 unlikely to be handled correctly.
- 428 ■ They fold on mathematical content. Usually, anything set in mathmode is recognized
429 and accounted for as one “formula”, regardless of whether it is a single character variable
430 or a complicated expression that fills a whole line. This is probably fine if the amount of
431 content in mathmode is only a very small fraction of the overall content. However, this
432 does not hold for a typical SoCG paper.

433 Wordcount achieves a greater separation between content and typographic layout com-
434 pared to linecount. However, we did not find a suitable tool that would make wordcount
435 practically feasible. To allow for a correct macro processing, such a tool would probably have
436 to be written in \LaTeX itself. Independently, the fundamental question of how to measure
437 the contents of mathematical expressions needs to be addressed.

438 Therefore, for the time being, linecount seems to be the more realistic option. There are
439 some technical issues with `lineno`, which does not assess certain environments correctly. But
440 these seem comparatively minor and easy to resolve. A line of text in a LIPICs document is
441 quite well defined: the fontsize is fixed, and textwidth does not vary between Letter and A4
442 settings. The separation between content and typographic layout is mostly with respect to
443 the vertical dimension, but that is a start. Also, we achieve an independent accounting for
444 figures and frontmatter, just by moving away from pagecount.

445 **Summary.** Moving from pagecount to linecount grants authors additional freedom of how
446 to present their results. It is much easier to justify the inclusion of graphical overviews and
447 examples, and the decision between inline and displaystyle representation of mathematical
448 content is much less driven by space considerations. Nobody will know about negative vspaces
449 anymore, nor understand why one would use `\noindent\textbf` instead of `\subparagraph`.
450 We trust authors to use this new flexibility to their and their reader’s advantage.

451 **Risks and Challenges.** If figures do not count, will their number and size grow beyond
452 reasonable? We are willing to trust the authors in this regard. If many figures make the
453 paper better, then they are welcome. If their number and/or size increases beyond reasonable,
454 reviewers will count this against the paper. So the incentive to go that way should be limited.
455 Something similar could be said for references: if they do not count, people could write
456 papers with 20 pages of references. If this really remains (or turns out to be) a concern, we
457 could, for instance, introduce a separate bound for the amount of figures.

458 Is counting lines too fiddly? Certainly, nobody wants to count lines by hand. An
459 automatic tool to do the counting is essential. After some testing (many thanks to Wouter
460 Meulemans, the Proceedings Chair of SoCG 2017, who checked with last year’s final versions),
461 the `lineno` package seems up to the task. But, of course, it is impossible to predict exactly
462 what issues people may run into with possibly highly customized personal environments. We
463 will have to see and then assess.

464 ——— References ———

- 465 1 Stephan I. Böttcher. `lineno.sty`—users manual version 3.1. [http://mirrors.ctan.org/
466 macros/latex/contrib/lineno/ulineno.pdf](http://mirrors.ctan.org/macros/latex/contrib/lineno/ulineno.pdf), 2001.

- 467 2 Rogério Brito. The algorithms bundle. [http://mirrors.ctan.org/macros/latex/contrib/](http://mirrors.ctan.org/macros/latex/contrib/algorithms/algorithms.pdf)
468 [algorithms/algorithms.pdf](http://mirrors.ctan.org/macros/latex/contrib/algorithms/algorithms.pdf), 2009.
- 469 3 Christophe Fiorio. `algorithm2e.sty`—package for algorithms release 5.2. [http://mirrors.](http://mirrors.ctan.org/macros/latex/contrib/algorithm2e/doc/algorithm2e.pdf)
470 [ctan.org/macros/latex/contrib/algorithm2e/doc/algorithm2e.pdf](http://mirrors.ctan.org/macros/latex/contrib/algorithm2e/doc/algorithm2e.pdf), 2017.
- 471 4 Dagstuhl Publishing. LIPICs: Instructions for authors and the lipics-v2019
472 class. [https://github.com/dagstuhl-publishing/styles/blob/v2019.3/LIPICs/authors/](https://github.com/dagstuhl-publishing/styles/blob/v2019.3/LIPICs/authors/lipics-v2019-authors-guidelines.pdf)
473 [lipics-v2019-authors-guidelines.pdf](https://github.com/dagstuhl-publishing/styles/blob/v2019.3/LIPICs/authors/lipics-v2019-authors-guidelines.pdf), 2020.
- 474 5 Axel Sommerfeldt. Customizing captions of floating environments. [http://mirrors.ctan.](http://mirrors.ctan.org/macros/latex/contrib/caption/caption-eng.pdf)
475 [org/macros/latex/contrib/caption/caption-eng.pdf](http://mirrors.ctan.org/macros/latex/contrib/caption/caption-eng.pdf), 2020. version 3.5.
- 476 6 Thomas F. Sturm. `tcolorbox`—manual for version 4.42. [http://mirrors.ctan.org/macros/](http://mirrors.ctan.org/macros/latex/contrib/tcolorbox/tcolorbox.pdf)
477 [latex/contrib/tcolorbox/tcolorbox.pdf](http://mirrors.ctan.org/macros/latex/contrib/tcolorbox/tcolorbox.pdf), 2020.
- 478 7 János Szász. The `algorithmicx` package. [http://mirrors.ctan.org/macros/latex/](http://mirrors.ctan.org/macros/latex/contrib/algorithmicx/algorithmicx.pdf)
479 [contrib/algorithmicx/algorithmicx.pdf](http://mirrors.ctan.org/macros/latex/contrib/algorithmicx/algorithmicx.pdf), 2005.
- 480 8 Mark Trettin and Jürgen Fenn. An essential guide to \LaTeX 2 ϵ usage. [http://mirrors.ctan.](http://mirrors.ctan.org/info/l2tabu/english/l2tabuen.pdf)
481 [org/info/l2tabu/english/l2tabuen.pdf](http://mirrors.ctan.org/info/l2tabu/english/l2tabuen.pdf), 2007.