

# On Counting Lines rather than Pages

## The CG Steering Committee and the SoCG 2019 PC Chairs

Gill Barequet, Mark de Berg, Erin Chambers, Michael Hoffmann, Joseph S.B. Mitchell,  
Bettina Speckmann, Monique Teillaud, Yusu Wang  
[socg-sc-18-20@inria.fr](mailto:socg-sc-18-20@inria.fr)

### 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.

For 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<sup>A</sup>T<sub>E</sub>X-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.

**2012 ACM Subject Classification** General and reference → General literature

**Keywords and phrases** `lineno`, `socg-lipics-v2019`

**Lines** 413

## 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<sup>A</sup>T<sub>E</sub>X-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` does and—to some extent—how it works. Section 4 describes how to add line numbering to



© Gill Barequet, Mark de Berg, Erin Chambers, Michael Hoffmann, Joseph S.B. Mitchell,  
Bettina Speckmann, Monique Teillaud, Yusu Wang;  
licensed under Creative Commons License CC-BY

February 21, 2019.



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

## XX:2 On Counting Lines rather than Pages

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.

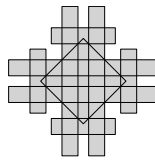
54 **Tables, Footnotes, etc.** Similar to figures, `lineno` does not handle tables and footnotes  
55 by default. But they should be counted, just as everything else. As an exception, tables do  
56 **not** count if they contain data only (usually, numbers). But they should be counted if they  
57 contain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

58 **Final version.** (New) The line limit also applies to the final version of the paper. Therefore,  
59 also the final version must be submitted with proper line numbering so that the proceedings  
60 chair 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`.



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

### 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]{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.

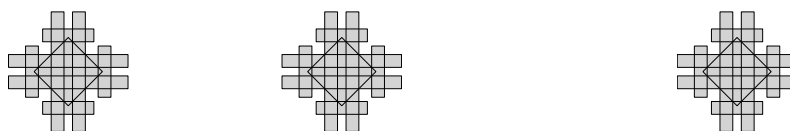
105 Due to the way  $\TeX$  handles floats, numbering them is tricky because their position in  
 106 the input may differ from their position in the output. The line numbers assigned by `lineno`  
 107 correspond to the spot where the figure appears in the input. For instance, Figure 1 appears  
 108 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

## XX: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, `soeg-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  
 123 that spans several lines.

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

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

```
131 \begin{tabular}{|c|c|c|}\hline
132   1 & happy & line \\ \hline
133   2 & happy & line \\ \hline
134 \end{tabular}
131 \begin{edtable}{tabular}{|c|c|c|}\hline
132   1 & happy & line \\ \hline
133   2 & happy & line \\ \hline
134 \end{edtable}
```

135 ■ **Table 1** L<sup>A</sup>T<sub>E</sub>X-code for a table using `tabular` in the original version (left) and wrapped into an  
 136 `edtable` (right).

137

1	happy	line
2	happy	line

138

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

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

148 Similar to captions, the numbering with respect to footnotes is not consecutive along the  
 149 page. While for captions this often can be fixed by moving the corresponding figure, table,

---

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

150 or caption in the source file, this does not really work for footnotes.<sup>2</sup> But at least we obtain  
151 a unique line number and a correct overall count.

152 **Arrays.** By default, arrays and its relatives are numbered as a single line. This is fine in  
153 many cases, for instance, where a matrix appears as a single entity, or if there are a few lines  
154 only that are sparsely filled compared to a full line of text. The definition in Line 156 below  
155 is such an example.

$$156 \quad f(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ is even;} \\ 3n + 1 & \text{if } n \text{ is odd.} \end{cases}$$

157 But in other cases, the amount and/or density of content in such a structure may not  
158 be appropriately accounted for by a single line of text. In such a case, the authors should  
159 number the lines. There are different ways to achieve this. For instance, the `amsmath`  
160 environments `align`, `flalign`, `gather` and `alignat`, along with their starred variants, are  
161 properly numbered by `lineno`; see the example in lines 164–166 below. (The example is  
162 short and sparse still, as the text would easily fit into a single line. It is intended to illustrate  
163 the numbering only, not a desparate need for it due to the excessive amount of content.)

$$164 \quad y = x + 2$$

$$165 \quad z \geq x - 1$$

$$166 \quad f(x, y, z) = x + y + z$$

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

```
169 \begin{edtable}[$$]{array}{rcl}
170   y & = & x+2 \\
171   z & \geq & x-1 \\
172 \end{edtable}
```

173 generates the following output.

$$174 \quad y = x + 2$$

$$175 \quad z \geq x - 1$$

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

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

---

146 <sup>2</sup> There is a package `fnlineno` that supports numbering footnotes properly. Alas, it only works for  
147 pagewise numbering and it does not seem to cooperate well with `\internallinenumbers`.

## XX:6 On Counting Lines rather than Pages

---

185 **Algorithm 1** Example using the `algorithms` package.

---

186 **Require:**  $n \geq 0$   
187 **Ensure:**  $n = 0$   
188     **while**  $N \neq 0$  **do**  
189         **if**  $N$  is even **then**  
190              $N \leftarrow N/2$   
191         **else**  $\{N$  is odd $\}$   
192              $N \leftarrow N - 1$   
193         **end if**  
194     **end while**

---

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

---

204 **Algorithm 2** Example pseudocode using `algorithm2e`.

---

205 **Data:** some input  
206 **Result:** some output  
207 **while** *not done* **do**  
208     compute some stuff;  
209     **if** *something happens* **then**  
210         do this;  
211     **else**  
212         do something else;  
213     **end**  
214 **end**

---

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

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

## 223 4 How to (Maybe) Handle Custom Environments

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

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

233 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.

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

237 The text in this box is put into a `minipage`, surrounded by an `fbox`. Using  
 240 238 `\internallinenumbers`, we obtain a proper numbering. But the outer line is still  
 239 numbered, resulting in a double numbering.

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

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

## 248 5 Specific Questions & Issues

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

### 251 5.1 L<sup>A</sup>T<sub>E</sub>X Error: File ‘lipics-v2019.cls’ not found

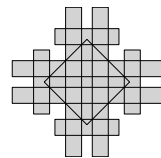
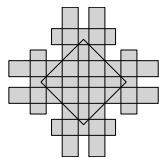
252 As described in Section 2, you need both, the `lipics-v2019` authors package and the  
 253 `socg-lipics-v2019` class file. The error message indicates that you have not installed the  
 254 files from the `lipics-v2019` authors package into the right location.

### 255 5.2 Zero Lines

256 If even after multiple runs of L<sup>A</sup>T<sub>E</sub>X, the “Lines” entry on the titlepage remains zero, then it  
 257 is most likely because you do not have a bibliography. Putting a bibliography command and  
 258 running `bibtex` should fix this problem. The reason is that the “Lines” entry is computed  
 259 from the start of the bibliography because everything following from that point on should  
 260 not count anymore. As a result, the entry is meaningful only if there is a bibliography.

261 **5.3 Figures Side by Side**

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



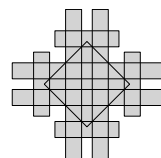
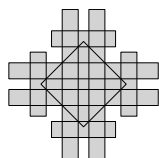
264 ■ **Figure 3** This caption spans several lines  
 265 that are numbered correctly.

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

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

```
273 \captionsetup{textformat=simple}
```

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



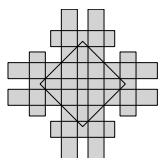
276 ■ **Figure 5** This caption spans several lines  
 277 that are numbered correctly. These line num-  
 278 bers are implicitly shared with Figure 6.

279 ■ **Figure 6** The lines in this caption are not  
 280 numbered, implicitly reusing the line numbers  
 281 from Figure 5.

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

281 **5.4 The Last Line of a Paragraph is not Numbered**

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



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

285 To avoid such effects, always separate floats from the surrounding text by properly ending  
 286 and starting the corresponding paragraphs, for instance, by leaving an empty line in between.  
 287 This was not done for the paragraph above, as its source code shown below reveals.  
 288



```

289 [...]
290 there is a spurious line number right after \figurename~\ref{fig:7}.
291 \begin{figure}[htbp]
292 [...]

```

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

## 295 5.5 Weird Line Number Placements

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

<pre> 299 300 301 302 303 </pre>	<p>The text in this box ... uses <code>\internallinenumbers</code>.</p> $\sum_{i=1}^n i^2 = \dots$ <p>There are too many numbers and they are not placed correctly.</p>
----------------------------------	---

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

```

313 \begin{nolinenumbers}
314   \begin{center}
315     \noindent\cprotect\fbbox{%
316       \noindent\begin{minipage}{.9\hsize}
317         \socgln The text in this box is numbered manually using \verb|\socgln|.
318         \[
319           ~\socgln\sum_{i=1}^ni^2 =\ldots
320         \]
321         \begin{minipage}{\hsize}\internallinenumbers
322           This paragraph consists of a longer text that is typeset using lines of
323           fixed height. It is wrapped into a nested minipage and collectively
324           numbered using \verb|\internallinenumbers|.
325         \end{minipage}
326       \end{minipage}
327     }
328   \end{center}
329 \end{nolinenumbers}

```

330 The resulting layout is given below.

## XX:10 On Counting Lines rather than Pages

331 The text in this box is numbered manually using `\socgnl`.

332 
$$\sum_{i=1}^n i^2 = \dots$$

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

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

### 338 **6 Why?**

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

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

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

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

361 The overarching goal is to measure the amount of content in a way that is independent  
362 from the typographical layout. This separation between content and typography is a main  
363 strength of a system like  $\text{\LaTeX}$ .

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

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

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

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

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

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

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

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

#### 414 ——— References ———

- 415 1 Stephan I. Böttcher. `lineno.sty`—users manual version 3.1. [http://mirrors.ctan.org/  
 416 macros/latex/contrib/lineno/ulineno.pdf](http://mirrors.ctan.org/macros/latex/contrib/lineno/ulineno.pdf), 2001.

## XX:12 On Counting Lines rather than Pages

- 417 2 Rogério Brito. The algorithms bundle. <http://mirrors.ctan.org/macros/latex/contrib/algorithms/algorithms.pdf>, 2009.
- 418
- 419 3 Christophe Fiorio. `algorithm2e.sty`—package for algorithms release 5.2. <http://mirrors.ctan.org/macros/latex/contrib/algorithm2e/doc/algorithm2e.pdf>, 2017.
- 420
- 421 4 Dagstuhl Publishing. The `lipics-v2018` class. <http://drops.dagstuhl.de/styles/lipics-v2018/lipics-v2018-authors/lipics-v2018-manual.pdf>, 2018.
- 422
- 423 5 Axel Sommerfeldt. Customizing captions of floating environments. <http://mirrors.ctan.org/macros/latex/contrib/caption/caption-eng.pdf>, 2018.
- 424
- 425 6 Thomas F. Sturm. `tcolorbox`—manual for version 4.14. <http://mirrors.ctan.org/macros/latex/contrib/tcolorbox/tcolorbox.pdf>, 2018.
- 426
- 427 7 János Szász. The `algorithmicx` package. <http://mirrors.ctan.org/macros/latex/contrib/algorithmicx/algorithmicx.pdf>, 2005.
- 428
- 429 8 Mark Trettin and Jürgen Fenn. An essential guide to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> usage. <http://mirrors.ctan.org/info/l2tabu/english/l2tabuen.pdf>, 2007.
- 430