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

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-v2018` LaTeX-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 classified :)

Keywords and phrases `lineno`, `socg-lipics-v2018`

Lines 410

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 LaTeX-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-v2018` around the `lipics-v2018` 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-v2018` does and—to some extent—how it works. Section 4 describes how to add line numbering to a nested `minipage` environment, which may be helpful as a blueprint if someone wants to extend `lineno` numbering to some custom environment. In Section 5 we list a few issues that authors may run into and what can be done about it (or not). Finally, in Section 6 we discuss our reasoning for the change to count lines rather than pages and summarize the results from our discussions.

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

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

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

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

2 What should Authors Do?

In short, there are three things:

(1) Get the `lipics-v2018` authors package from LIPIcs [4] and the `socg-lipics-v2018` class file from the Computational Geometry Pages. Note that the latest LIPIcs class (`lipics-v2018` from April 9, 2018) is needed, earlier versions do not work.
If you want to start from a template, use the sample article file from the LIPIcs authors package, but replace the document class `lipics-v2018` by `socg-lipics-v2018`.

(2) Use the `socg-lipics-v2018` wrapper class around the standard `lipics-v2018` document class as your main document class. It attempts to provide a more consistent line numbering by fixing issues with various command and environments. The next section goes in some detail over the different issues addressed by this wrapper. Some features can be switched off separately, in case they give trouble.

(3) Do **not** use `$$ ... $$` to typeset displaymath formulae! Use `\[ ... \]` instead! You will find various arguments for this advice on the Internet (see, for instance l2tabu [8]), but our main reason here is that the plain-TEX primitive `$$` does not work well with `lineno`.

3 What Does the `socg-lipics-v2018` Class Do?

In this section we list the different tweaks that the `socg-lipics-v2018` wrapper class applies to the standard `lipics-v2018` document class. This is intended mostly as a documentation for those who are interested to know exactly what happens. It should also help people to figure out what goes wrong in case of problems, to handle their custom macros in a similar

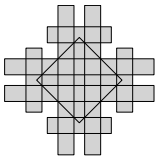


Figure 1 This figure is placed at the top of the page. But the caption line numbers correspond to the place where it appears in the input LaTeX-file.

fashion, or to suggest improvements. In order to use this class, you do not necessarily need to read through this section, you can simply use, for instance,

```
\documentclass[a4paper,USenglish]{socg-lipics-v2018}
```

and see what happens.

Frontmatter and Bibliography. To disregard the frontmatter, `socg-lipics-v2018` disables line numbering there by replacing the `\maketitle` command. Also the lines that contain subject classifications, etc. are considered part of the frontmatter and, therefore, not numbered. The DOI entry is disabled. An entry *Lines* showing the number of lines in the paper is shown instead. It is computed from the linenumber where the bibliography starts. As for the bibliography, it does not hurt to leave the numbers there so that reviewers can refer to them. Just remember that these lines do not count toward the 500 lines bound.

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

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

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

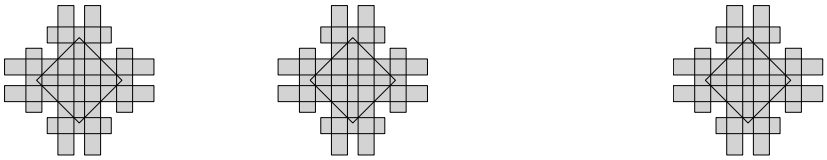
At least the map : output line $\to$ line number is injective and the overall count works out. Unless LaTeX places the figure in the middle of a paragraph, the line numbering can be made consecutive by moving the figure in the input to the position where it appears in the output. (This is nice to have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

Footnotes. Similar to floats, footnotes are tricky because their placement is determined at the end of a page only, and it usually differs from their position in the input. By default, `lineno` does not number them. In order to fix this, `socg-lipics-v2018` wraps the contents of every footnote into a `minipage`, which is then numbered using `\internallinenumbers`.[1]

[1] This is an insightful footnote. Of course, we want it to get a proper line number. The number is "stolen" from the beginning of the paragraph where the footnote is referenced, not inserted at the end of the page.

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

Subcaptions. The `caption` package offers a `\subcaption` command to combine several sub-figures into one single figure. While such an aggregation should not be necessary anymore just to save page-space, it is still useful as a means to structure a collection of related figures. By default, `socg-lipics-v2018` numbers all subcaptions, using the same line number(s) for subcaptions that appear along the same output line. Also the `subfigure` environment is handled accordingly. Figure 2 below shows an example. In case this feature gives trouble, it can be switched off using the documentclass option `nosubfigcap`.



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

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

Tables. The fix described above for figures addresses all captions. For the actual table contents, the `edtable` package is convenient. To get proper line numbers for a standard table environment such as `tabular`, it can be wrapped into into an `edtable`. For instance, the code in Table 1 (left) is effectively processed as shown in Table 1 (right) to appear in the output as shown in Table 2. By default, `socg-lipics-v2018` wraps all `tabular` environments into an `edtable`. To globally disable this wrapping, use the documentclass option `notab`.

```
\begin{tabular}{|c|c|c|}\hline
  1 & happy & line \\\hline
  2 & happy & line \\\hline
\end{tabular}
```

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

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

1	happy	line
2	happy	line

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

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

Arrays. By default, arrays and its relatives are numbered as a single line. This is fine in many cases, for instance, where a matrix appears as a single entity, or if there are a few lines only that are sparsely filled compared to a full line of text. The definition in Line 151 below is such an example.

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

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

$$\begin{aligned} y &= x+2 \\ z &\geq x-1 \\ f(x,y,z) &= x+y+z \end{aligned}$$

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

```
\begin{edtable}[$$]{array}{rcl}
  y & = & x+2 \\
  z & \ge & x-1 \\
\end{edtable}
```

generates the following output.

$$\begin{array}{rcl} y & = & x+2 \\ z & \geq & x-1 \end{array}$$

However, this wrapping does not work from within mathmode, which makes it a bit clumsy to use. Therefore, `socg-lipics-v2018` does not perform any automatic wrapping of `array`s.

Algorithms. Both the `algorithms` package [2] and the `algorithmicx` package [7] provide two environments `algorithmic` and `algorithm` to format pseudocode. The class `socg-lipics-v2018` adds line numbers to captions and to the `algorithmic` environment using `\internallinenumbers`. See Algorithm 1 below for an example using the `algorithms` package. This feature is enabled only if the packages `algorithm` and `algorithmic(x)`, respectively, are loaded in the preamble of the document. It can be disabled with the documentclass option `noalgorithms`.

Algorithm 1 Example using the `algorithms` package.

Require: $n \geq 0$
Ensure: $n = 0$
  while $N \neq 0$ **do**
    if N is even **then**
      $N \leftarrow N/2$
    else {N is odd}
      $N \leftarrow N - 1$
    end if
  end while

Algorithm2e. The `algorithm2e` package [3] provides an environment to format pseudocode. It has its own version of many standard macros, such as line numbers and captions. Its customization options do not seem powerful enough to achieve a style that is consistent with both LIPIcs and `lineno`. Therefore, `socg-lipics-v2018` changes some internal macros of `algorithm2e` so as to (1) obtain linenumbers for both the code (using `algorithm2e`'s own numbering option) and the caption (using `lineno`) and (2) change the appearance to fit with LIPIcs and `lineno`. Algorithm 2 below illustrates a resulting layout. This feature is enabled only if the package `algorithm2e` is loaded in the preamble of the document. It can be disabled with the documentclass option `noalgorithm2e`.

Algorithm 2 Example pseudocode using `algorithm2e`.

Data: some input
Result: some output
while *not done* **do**
  compute some stuff;
  if *something happens* **then**
    do this;
  else
    do something else;
  end
end

Tcolorbox. The `tcolorbox` package [6] provides an environment for colored and framed text boxes. The `socg-lipics-v2018` class handles these environments by adding the command `\internallinenumbers` and adjusting the spacing to avoid overlap between line numbers and the surrounding box. This feature is enabled only if the package `tcolorbox` is loaded in the preamble of the document. It can be disabled with the documentclass option `notcolorbox`.

This text is wrapped into `\begin{tcolorbox}` ... `\end{tcolorbox}`. Such a simple example is handled fine by `socg-lipics-v2018`. If you work with more elaborate custom boxes, you may have to do some manual tuning yourself.

4 How to (Maybe) Handle Custom Environments

The `socg-lipics-v2018` class attempts to handle a number of frequently occurring issues with `lineno`. But, depending on what packages and macros people use, they may run into issues that are not covered there. In such a case, it makes sense to check whether there is an easy workaround with some minor amount of manual tweaking. As a typical example let us consider the `minipage` environment because (1) it can be easily adopted to get some line numbers going and (2) it can be used as a tool to handle other issues, by wrapping contents into a minipage. In essence, this is what most of the fixes in `socg-lipics-v2018` do.

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

> 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.

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

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

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

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

5 Specific Questions & Issues

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

5.1 LaTeX Error: File 'lipics-v2018.cls' not found

As described in Section 2, you need both, the `lipics-v2018` authors package and the `socg-lipics-v2018` class file. The error message indicates that you have not installed the files from the `lipics-v2018` authors package into the right location.

5.2 Zero Lines

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

5.3 Figures Side by Side

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

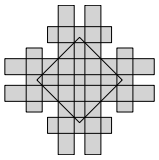


Figure 3 This caption spans several lines that are numbered correctly.

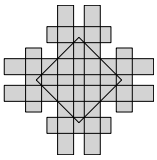


Figure 4 The lines in this caption are also numbered, leading to an overcount.

In order to avoid this, we would like to number the lines of the longest of these captions only. Fortunately, it is easy to switch off line numbering for a single caption. The class `socg-lipics-v2018` implements line numbering using a customization option of the `caption` package [5]. More precisely, it defines a `textformat` called `socgnumberitall` and sets this to be the default. So, placing the command

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

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

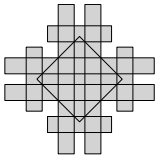


Figure 5 This caption spans several lines that are numbered correctly. These line numbers are implicitly shared with Figure 6.

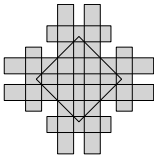


Figure 6 The lines in this caption are not numbered, implicitly reusing the line numbers from Figure 5.

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

5.4 The Last Line of a Paragraph is not Numbered

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

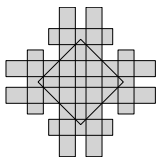


Figure 7 A figure may disturb the line numbering for the previous paragraph if it is not properly separated from that paragraph.

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

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

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

5.5 Weird Line Number Placements

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

The text in this box ... uses `\internallinenumbers`.

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

There are too many numbers and they are not placed correctly.

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

```
\begin{nolinenumbers}
  \begin{center}
    \noindent\cprotect\fbox{%
      \noindent\begin{minipage}{.9\hsize}
        \socgnl The text in this box is numbered manually using \verb|\socgnl|.
        \[
          ~\socgnl\sum_{i=1}^ni^2 =\ldots
        \]
        \begin{minipage}{\hsize}\internallinenumbers
          This paragraph consists of a longer text that is typeset using lines of
          fixed height. It is wrapped into a nested minipage and collectively
          numbered using \verb|\internallinenumbers|.
        \end{minipage}
      \end{minipage}
    }
  \end{center}
\end{nolinenumbers}
```

The resulting layout is given below.

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

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

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

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

6 Why?

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

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

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

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

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

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

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

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

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

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

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

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

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

References

1. Stephan I. Böttcher. `lineno.sty`—users manual version 3.1. http://mirrors.ctan.org/macros/latex/contrib/lineno/ulineno.pdf, 2001.
2. Rogério Brito. The algorithms bundle. http://mirrors.ctan.org/macros/latex/contrib/algorithms/algorithms.pdf, 2009.
3. Christophe Fiorio. `algorithm2e.sty`—package for algorithms release 5.2. http://mirrors.ctan.org/macros/latex/contrib/algorithm2e/doc/algorithm2e.pdf, 2017.
4. Dagstuhl Publishing. The `lipics-v2018` class. http://drops.dagstuhl.de/styles/lipics-v2018/lipics-v2018-authors/lipics-v2018-manual.pdf, 2018.
5. Axel Sommerfeldt. Customizing captions of floating environments. http://mirrors.ctan.org/macros/latex/contrib/caption/caption-eng.pdf, 2018.
6. Thomas F. Sturm. `tcolorbox`—manual for version 4.14. http://mirrors.ctan.org/macros/latex/contrib/tcolorbox/tcolorbox.pdf, 2018.
7. János Szász. The `algorithmicx` package. http://mirrors.ctan.org/macros/latex/contrib/algorithmicx/algorithmicx.pdf, 2005.
8. Mark Trettin and Jürgen Fenn. An essential guide to LaTeX 2_ε usage. http://mirrors.ctan.org/info/l2tabu/english/l2tabuen.pdf, 2007.