Managing forlorn paragraph lines (a.k.a. widows and orphans) in LaTeX

Frank Mittelbach

Contents

1 The name of the game

Splitting off the first or last line of a paragraph at a page or column break is considered bad practice in typesetting circles. It is thus not surprising that the craftspeople have come up with fairly descriptive names for such lines when they appear in typeset documents.

Commonly used are the terms "widow" for the last and "orphan" for the first line. These are, for example used in English, French ("veuve" and "orpheline"), Italian ("Vedova" and "Orfano"), Spanish ("línea huérfana" and "línea viuda"), or to a lesser extent in German ("Witwe" and "Waise").

One way to remember them is to think of orphaned lines appearing at the start (birth) and widows near the end (death) of a paragraph or by using Bringhurst's mnemonic, "An orphan has no past; a widow has no future" [6].

German typesetters coined some more profane descriptions by calling the widow line a "Hurenkind" (child of a whore) and the orphan line a "Schusterjunge" (son of a shoemaker) allegedly because these boys have been notoriously meddlesome. For German practitioners these are still the predominantly used terms, though "Witwen" and "Waisen" are also well understood. Dutch uses "hoerenjong" and "weeskind" which translates to son of a whore and orphan, i.e., somewhere in between the German usage and the other languages.

Don Knuth catered for this typographic detail in the TeX program by providing parameters whose values are used as penalties if the pagination algorithm considers breaking in such a place. Widow lines are penalized via `\widowpenalty`; however, orphans are not controlled by `\orphanpenalty` as one might expect, but by a parameter named `\clubpenalty`.

There have been some queries about this choice of names on Stack Exchange and after a little Internet searching I found a listing for "club line" in the Collins English Dictionary Digital Edition [1], listing it as a British (!) term used in printing for an orphan line. Further checks through the first dozen or so pages of google hits for "club line" by its own, and the same with additional restrictions such as "printing" or "typography" revealed a handful of additional references (two of which mentioned that Knuth used the term — thus circular references). So on the whole it was a meager result and all except one indicated a use of the term in British not American English.

Contrast this with a search for "orphan line" in google: Now we will find that nearly all the results in the first five pages are relevant, with only two or three near the end being unrelated.

However, what we also see from following them up is that about a third of them give the terms different meanings, either by swapping the definition of orphan and widow lines or by giving them a slightly different meaning altogether: The last line of a paragraph being nearly empty, i.e., containing only a single word or even part of a single word.[1]

Most of the time, though, the forlorn lines we want to deal with are called widows and orphans and this is what we will call them in the remainder of the article, even if we have to set a `\clubpenalty` to deal with one of them.

2 The problem

Essentially everyone in typography circles agrees that widows and orphans are very distracting to the reader as well as a sign of bad craftsmanship, and should therefore be avoided. In fact, most writing guides and other books on typography generally suggest that a document should have no such lines whatsoever, e.g., in older editions of the Chicago Manual of Style [2] we find "A page should not begin with the last line of a paragraph unless it is full measure and should not end with the first line of a new paragraph."

However, that is easier said than done, so in a newer edition of that guide [4] we now find "A page should not begin with the last line of a paragraph unless it is full measure. (A page can, however, end with the first line of a new paragraph.)" instead.

As a result of this sort of guidance many journal classes for LaTeX completely forbid widows and orphans by setting `\widowpenalty` and `\clubpenalty` to `10000` which prohibits a break at such points — *TUGboat* being no exception.

[1] ... as demonstrated here. In TeX this kind of typographical issue can also be dealt with, although by different means and somewhat more manually: The parameter `\finalhyphenpenalty` can make hyphenation in the last line unattractive, and using unbreakable spaces will ensure that there is more than one word in the last line; `\parfillskip` can also help.

But doing this introduces severe problems: As LaTeX (and in fact all major typesetting systems to date) use a greedy algorithm to determine the pagination of a document, it will recognize problems with orphan or widow lines late in the game and will have only the current page to work with. This means the best it can do to avoid the situation is to push an orphan to the next page if there is not enough room to squeeze in another line. The same happens with widows; here LaTeX is forced to move the second-last line to the next page even though it would still nicely fit.

As a result the current page will have an additional line-height worth of white space that needs to be distributed somewhere on the page. If there are headings, displays, lists or other objects for which the design allows some flexibility in the surrounding white space, then this extra space may not create much of an issue. If, however, the page consists only of text or objects without any flexibility, then all LaTeX can do is run the page or column short, generating a fairly ugly hole at the bottom.[2]

Related problems

Besides widows and orphans there are a number of similar issues that typography manuals mandate eliminating if at all possible. One is a paragraph split across pages at a hyphenation point so that only a part of the word is visible at any time; another is a widow line with a following display formula. For both, TeX offers parameters to control the undesirability of the scenario. By default Don Knuth considered them of lesser importance and provided default values of `100` and `50` for `\brokenpenalty` and `\displaywidowpenalty`, respectively while he specified `150` for orphans and widows. However, if your style guide (or your class file) wants to avoid them at all cost then you are in precisely the same situation as with widows and orphans discussed above.

A special variation of the last issue is a display formula starting a page, that is, with the introductory material completely on the previous page or column. That is considered a no-go by nearly everybody so in TeX the controlling `\predisplaypenalty` parameter has by default a value of `10000`. But again, there may be valid reasons to ignore this advice in a special situation, e.g., when the space constraints are high.

3 Fixing the problem

The alternative to preventing widows and orphans (or hyphens across page boundaries, etc.) automatically and at all costs is to manually resolve the issues when they arise. For this one finds a number of suggestions in the typography literature; a good collection is given in the guidelines section of the Wikipedia page on "Widows and Orphans" [5]. We will look at them one by one below and discuss their applicability and possible implementation in a LaTeX document.

- ▶ *Forcing a page break early, producing a shorter page*

This is what LaTeX and most other typesetting tools automatically do if you completely forbid widows and orphans and if often leads to badly filled pages as discussed above.

However, if you force the page break manually, you can lessen the impact by also explicitly forcing earlier breaks and thereby shifting the extra whitespace to a page or column where it can be absorbed by the available flexibility on that page.

- ▶ *Adjusting the leading, the space between lines of text (although such carding or feathering is usually frowned upon)*

That is indeed frowned upon and for good reason. The human eye is tuned to notice even small differences in the vertical spacing of lines and across columns or pages such changes, even if they are small, are very noticeable and distracting. Besides, managing such a change in a LaTeX document would be, while possible, quite cumbersome, so this is not particularly useful advice for us.

- ▶ *Adjusting the spacing between words to produce 'tighter' or 'looser' paragraphs*

This is certainly a practical option if you choose the right paragraph or paragraphs, e.g., those that are somewhat longer and that have a last line that is either nearly full (for lengthening) or nearly empty (for shortening). In that case squeezing the word spaces might result in one line less and extending it might get you an additional line (with just a word or two). In many cases the resulting gray value is still of acceptable quality so this is a typical trick of the trade. In LaTeX this is achieved by using the `\looseness` parameter that is discussed in Section 4.

Note that you do not necessarily need to manipulate one of the paragraphs of the problem page; there might be a better candidate on an earlier page.

- ▶ *Adjusting hyphenation within the paragraph*

This is a variation of the "change the number of paragraph lines" type of approach. However, given that LaTeX is usually good in considering most of the possible hyphenation points when breaking paragraphs, one is unlikely to gain much if anything. Thus, this suggestion might have some merits in a system that

[2] This can be observed on the first page of this article, where an orphan line was pushed onto the current page. Fortunately, the resulting hole is partly masked by the footnote.

does not hyphenate well (or at all) but with LaTeX one is better off applying `\looseness`.

If you have a very badly broken paragraph because of a missed hyphenation point you should fix that anyway (by adding `\-` or a `\hyphenation` exception) regardless of whether or not there is a widow or orphan nearby.

► *Adjusting the page's margins*

Now this is an interesting approach. In contrast to changes in `\baselineskip` small changes in the line width are virtually undetectable without a ruler — unless you make it a huge change. Unfortunately, it is not really an option when using LaTeX as the underlying TeX engine essentially assumes a fixed line width throughout, so it is fairly difficult to change that at arbitrary places.

In other words, this advice is really geared towards interactive systems where you can change the width of a text region and get an immediate reflow as a visual feedback.

► *Subtle scaling of the page, though too much non-uniform scaling can visibly distort the letters*

In principle, that would be possible with LaTeX though so far nobody has implemented the necessary changes to the output routine. That is, when a column or page runs short it will be scaled to the right height before attaching headers and footers. Instead of a non-uniform scaling, one could do uniform scaling and adjust the horizontal widths of headers and footers accordingly.

"An orphan has no past; a widow has no future"

However, that approach has issues already discussed: Scaling means we get a different leading (though this time the characters will also grow). And if we do non-linear scaling, i.e., only vertically, then we will distort characters and from a certain point onwards this will also be noticeable. So it is questionable whether this will actually improve the situation.

► *Rewriting a portion of the paragraph*

This is obviously something you can do only if you are the author and not typesetting some text written by others. But if so, it is a valid strategy since it enables you to easily shorten or enlarge a paragraph so that your orphan or widow is reunited with other lines.

Again, there is no requirement to do this rewrite with the paragraph causing the issue (as implied by the advice); you can choose any[3] earlier paragraph to achieve the desired effect.

[3] Well, "any" is an exaggeration: If you change a paragraph on an earlier page the gained (or extra) space might get swallowed up by available flexibility on some intermediate page and your widow or orphan thus stays put.

► *Reduce the tracking of the words*

Tracking in this context means adjusting the spacing between characters in a uniform way (in contrast to kerning, which means adjusting the spacing between individual glyph pairs, e.g., "AV" cf. "AV").

Figure 1 shows a line of text with different amounts of tracking (negative and positive) applied. Clearly by applying tracking one can shorten or lengthen a text. However, when comparing the lines side by side it is also obvious that the gray value of words changes fairly rapidly too. Thus even with small tracking values, changes may become noticeable and thus distracting. To illustrate the point this article contains one manipulated paragraph; see if you can spot it — perhaps it was on an earlier page and you thought: hmm that doesn't look quite right.[4]

On the whole, common typographical advice is to *not use* tracking for such purposes or, if there is no better alternative, then only with very small tracking values in which case there may not be any noticeable effects on the paragraph length unless you are lucky. It is possible to experiment in LaTeX if you load the microtype package and use, for example, `\textls`.

► *Adding a pull quote to the text (more common for magazines)*

Pull quotes are catch phrases from the text that are "pulled out" and typeset prominently again in a different place, typically in a larger and often different font. They serve as eye catchers and if carefully chosen will give the reader a preview of the content or main points of an article.

The design needs to clearly distinguish them from other display material, e.g., there should be no way to confuse them with headings, etc. In two-column texts this is often done by placing them in a window with both columns flowing around them (as shown on this page), but placing them into the content of one column is also often done.

Placing them within a single column is fairly easy in TeX, all you need to do is to define an environment that places the material between paragraph lines (using `\vadjust` if used inside paragraph text).

Producing pull quotes with the column texts flowing around it, is more manual work and fairly cumbersome, but doable. On the present page, we used the wrapfig package. The approach, as well as a few others are discussed in answers to a question on Stack Exchange [3].

[4] The answer is given at the end of the article.

Tracking is the uniform increase or decrease of spacing between glyphs.	−0.05 em
Tracking is the uniform increase or decrease of spacing between glyphs.	−0.03 em
Tracking is the uniform increase or decrease of spacing between glyphs.	−0.02 em
Tracking is the uniform increase or decrease of spacing between glyphs.	−0.01 em
Tracking is the uniform increase or decrease of spacing between glyphs.	— LaTeX's default setting —
Tracking is the uniform increase or decrease of spacing between glyphs.	+0.01 em
Tracking is the uniform increase or decrease of spacing between glyphs.	+0.02 em
Tracking is the uniform increase or decrease of spacing between glyphs.	+0.05 em
Tracking is the uniform increase or decrease of spacing between glyphs.	+0.07 em

Figure 1: Tracking in action

As the above advice already mentions, these are more commonly found in magazine type documents, so this approach may or may not be applicable.

► *Adding a figure to the text, or resizing an existing figure*

Just like adding a pull quote, resizing a figure will obviously change the amount of material a column can hold and thus will enable us to move an orphan or widow out of harm's way. Whether or not it is a valid option depends on the figure in question; often enough graphics do offer some freedom and can be adjusted either by scaling (up or down) or by cropping, etc.

Summary

To resolve issues with widows and orphans one has to somehow adjust the amount of material typeset in the respective column or page. For LaTeX users the most promising approaches are

- forcing material from one column to the next through explicit page breaks;
- generating more or less material by lengthening or shortening some paragraphs;
- rewriting paragraphs (if you are the author);
- or resizing a float.

Adjusting the line width for a single column is rather difficult to achieve in LaTeX and therefore not recommended, even though it can lead to good results. Applying tracking seldom works well, it usually either makes no difference or results in noticeable grey-level differences.

Using pull quotes is similar to changing or resizing floats or modifying paragraphs. However, the quotes carry meaning and so you can't simply add one arbitrarily for the sake of better pagination. Thus, adding or moving them around is a bit like changing the document structure and you therefore have to be careful not to sacrifice semantics for form. This makes them a less desirable approach.

Scaling the page or changing the leading is typographically rather questionable, so these can't be recommended (besides their being rather complicated to achieve with LaTeX).

In any case, it should be noted though that all approaches are manual and thus the adjustments will become invalid the moment there is a document change that modifies the amount of material typeset. It is therefore of paramount importance to manually fix widows and orphans only at the very last stage of producing the final document. Otherwise all the effort might be in vain and will need to be undone or changed over and over again.

The situation would be somewhat different if TeX was extended to globally optimize pagination rather than applying a greedy algorithm as it currently does. Some theoretical work in that direction has been carried out in recent years by the current author and it may eventually lead to a production-ready system using LuaTeX [7, 8]. However, at present it is available only in a private prototype implementation and can't be used with vanilla LaTeX.

4 Using TeX's `\looseness` approach

TeX (and therefore LaTeX) uses a globally optimizing line-breaking algorithm to find the best breaks for a given paragraph based on a given set of parameters. One can ask TeX to try to find a solution (within given quality boundaries) that is a number of lines longer or shorter than the optimal result. If such a solution exists it will be used; if not, then TeX will try to match the request as closely as possible.

The paragraph will still be optimized (under the new conditions), i.e., its overall gray level will be fairly uniform, etc., but, inevitably, the inter-word spacing will get looser or tighter in the process.

To activate this feature you need to set the parameter `\looseness` to the desired value. This has to be done directly in front of (or within) the paragraph text via low-level TeX syntax

```
\looseness=1    % to lengthen by one line
%               % <- no blank line here!
The text that gets manipulated ...
```

as there is no LaTeX interface available.

TEX automatically resets the value to zero whenever a `\par` command or blank line is encountered; thus it will affect at most one paragraph.

A value of `-1` has the best chance to work if the last line is already nearly empty (and the paragraph is of reasonable length). Lengthening is somewhat easier as inter-word spaces can stretch arbitrarily (as long as they do not exceed the `\tolerance`), whereas they can shrink by only a fixed amount. But again there is a better chance for success if the last line is already (nearly) filled.

So far so easy, but there are a few pitfalls that need to be avoided: First of all, with a positive value of `\looseness` TEX will usually move only a single word or even part of a single word into the last line, as this way there is more material in the others and thus less stretching of the inter-word spaces necessary. As this usually looks rather ugly, it is best to tie the last words together by using `~` and if necessary prevent hyphenation of the last word by placing an `\mbox` around it.

Secondly, lengthening of a paragraph may go horribly wrong (as shown here) if the document is set with a high `\tolerance` value, e.g., most definitely when a `\sloppy` declaration is in force.

The `\tolerance` defines how bad a line can get while still being a candidate for the line-breaking algorithm and `\sloppy` (or `sloppypar`) simply sets this tolerance nearly[5] to infinity, i.e., arbitrarily bad lines are acceptable and thus you might end up with a paragraph like this one (where we asked for two extra lines and got them). With a lower `\tolerance` value that would never have happened: TEX would have refused to produce a result like this.

Seeing the previous paragraph you might ask yourself why one would want to use a high, let alone infinite, `\tolerance` at all. The reason is that this caters for situations where line breaking is very difficult. If there is a better solution with a lower tolerance value TEX would use it, but if not it could still proceed. So normally we wouldn't see such bad paragraphs even with a high tolerance in force (it just means that TEX evaluates more candidate solutions). But if we apply `\looseness` we explicitly ask TEX to deviate from the optimal number of lines and to fulfill this request TEX may resort to a solution with bad lines that nobody would want.

[5] In the early days of LATEX `\sloppy` used to set the tolerance to `10000` (i.e., TEX's infinity) but that tended to produce even more bizarre looking paragraphs: TEX then made one line really, really bad and all others perfect, as that looked to the optimizer to be the best solution.

5 Identifying pages with widows or orphans

If the document class you use sets `\widowpenalty` and `\clubpenalty` to `10000`, then LATEX will automatically prevent widows and orphans, i.e., an orphan is forced to the top of the next page or column; and the same with the line preceding a widow. The downside, as discussed previously, is partly empty pages and if space is a premium (for example, if your conference paper is not allowed to be more than X pages in total) then this is a possible problem. Thus you are better off allowing widows and orphans (by changing the parameter values) and manually correcting them in one way or another.

The question then becomes, how do you identify the problematic page breaks without manually going through the printout of your document and searching for them? While that is certainly an option it is error prone and it would be much nicer if LATEX (even if it can't automatically resolve the issues for you) at least identifies them so that you only have to check the problem pages.

This is possible by simply loading the package widows-and-orphans.[6] This package adjusts the parameter values slightly so that so that all possible combinations lead to distinctive numbers. For example, instead of the LATEX default values it would choose

```
\widowpenalty        = 150
\clubpenalty         = 152
\displaywidowpenalty =  50
\brokenpenalty       = 101
```

so that it can distinguish between a widow and an orphan (`\widowpenalty` or `\clubpenalty`) or a display widow that comes together with a hyphen at the break (`\displaywidowpenalty` + `\brokenpenalty`). In case you wonder why `151` wasn't used: that value is already used by LATEX for `\@medpenalty` which you get if you issue `\nopagebreak[2]`. By making sure that all technically possible combinations lead to unique numbers it is only necessary to look at the penalty of the page break to determine whether or not that break exhibits one or more of the problems. So at any page or column break the `\outputpenalty` is inspected and depending on the findings a warning or error is generated that can then be checked and corrected manually.

To ease this process further the package has a number of key/value options. The `check` option determines how findings are handled: the default

[6] The implementation of the package (which is written in the expl3 programming language) is documented in a separate article in this *TUGboat* issue [9].

is `warning` in which case warnings are written to the terminal and the `.log` file. In the last phase of document development you may want to change that to `error` in which case the package will stop at each problem with an error message rather than just a warning. In the opposite direction, `info` will not clutter the terminal with messages and only writes to the `.log`. And if you know for sure that all the remaining issues have to stay, you can also use the value `none` in which case no checks are done at all.

Why would one want the last option instead of not loading the package? The reason is this: as the package has to change the parameter settings slightly, not loading it would mean running the document with different values and even though those changes are minimal, it is possible to construct examples where the difference matters and leads to changed results. So once you have fixed what is possible to fix it's safest to still load the package, even if you no longer want the remaining warnings. Another reason is that the package offers the command `\WaOsetup` that allows you to change options mid-document, e.g., turn the warnings off for chapters already handled, but turn them on again for others.

Instead of suppressing all checking for a part of the document via `\WaOsetup` you can issue the command `\WaOignorenext` somewhere in the document, after which the next check — for the current page or column — will be silenced. The check is still performed and if no problems are found you will receive an error message, because either you have added it to the wrong page or your text has changed and it is no longer needed.

The package also offers options to set individual parameters to "reasonable" values. These are `widows`, `orphans`, `hyphens` that all accept `default` (LaTeX default), `avoid` (higher value but still possible) or `prevent`. And there are also the valueless options `default-all`, `avoid-all` and `prevent-all` to set all parameters in one go. Of course, as an alternative one can always change the parameters individually in the preamble or even in the middle of the document by assigning explicit numerical values.

If you want to see the resulting parameter settings (and the combinations that need to be unique in order to allow the package to work) you can issue the command `\WaOparameters` at any point after the preamble, which will give you a somewhat terse listing.

Answer to the riddle

The paragraph with negative tracking (−0.01 em) is the first one after "▶ Rewriting a portion . . . " on page 903, toward the bottom of the first column.

Due to the tracking it needs one line less compared to the default line breaks. But as a result of the tracking, the characters are noticeably closer to each other, for example, in the word "obviously". Depending on your aesthetic judgment, a value of ±0.02 em is roughly the borderline of what can be considered acceptable, so if that or a lower value works, it might be an option.

References

[1] Anonymous. Collins English dictionary — complete & unabridged 2012 digital edition. `https://www.collinsdictionary.com/dictionary/english/club-line`.

[2] Anonymous. *The Chicago Manual of Style*. University of Chicago Press, Chicago, IL, USA, 14th edition, 1993.

[3] Anonymous. How can you create pullquotes?, 2012. `https://tex.stackexchange.com/questions/45709/how-do-you-create-pull-quotes`.

[4] Anonymous. *The Chicago Manual of Style*. University of Chicago Press, Chicago, IL, USA, 17th edition, 2017.

[5] Anonymous. Widows and orphans, 2017. `https://en.wikipedia.org/wiki/Widows_and_orphans`.

[6] Robert Bringhurst. *The Elements of Typographic Style*. Hartley & Marks Publishers, Point Roberts, WA, USA and Vancouver, BC, Canada, 1992.

[7] Frank Mittelbach. A general framework for globally optimized pagination. In *Proceedings of the 2016 ACM Symposium on Document Engineering*, DocEng '16, pages 11–20, New York, NY, USA, 2016. ACM. Download from `https://www.latex-project.org/publications`.

[8] Frank Mittelbach. Effective floating strategies. In *Proceedings of the 2017 ACM Symposium on Document Engineering*, DocEng '17, pages 29–38, New York, NY, USA, 2017. ACM. Download from `https://www.latex-project.org/publications`.

[9] Frank Mittelbach. The widows-and-orphans package. *TUGboat* 39:3, 2018, 901–911. `https://ctan.org/pkg/widows-and-orphans`

◇ Frank Mittelbach
Mainz, Germany
`frank.mittelbach (at) latex-project dot org`
`https://www.latex-project.org`