Similarity Problems in Paragraph Justification

An Extension to the Knuth-Plass Algorithm

Didier Verna
EPITA Research Laboratory
Le Kremlin-Bicêtre, France
didier@lrde.epita.fr

ABSTRACT

In high quality typography, consecutive lines beginning or ending with the same word or sequence of characters is considered a defect. We have implemented an extension to TEX's paragraph justification algorithm which handles this problem. Experimentation shows that getting rid of similarities is both worth addressing and achievable. Our extension automates the detection and avoidance of similarities while leaving the ultimate decision to the professional typographer, thanks to a new adjustable cursor. The extension is simple and lightweight, making it a useful addition to production engines.

CCS CONCEPTS

• **Applied computing** → **Document preparation**; • **Theory of computation** → *Dynamic graph algorithms.*

KEYWORDS

Paragraph Justification, Similarity Avoidance, Homeoteleutons, Homeoarchies, TEX, Knuth-Plass Extension

ACM Reference Format:
Didier Verna. 2024. Similarity Problems in Paragraph Justification: An Extension to the Knuth-Plass Algorithm. In *ACM Symposium on Document Engineering 2024 (DocEng '24), August 20–23, 2024, San Jose, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3685650.3685666

1 INTRODUCTION

In spite of its relatively old age, Donald Knuth's TEX typesetting system [10, 11] is still considered a *de facto* standard when it comes to digital typography. In particular, its paragraph justification algorithm, known as the *Knuth-Plass* [12], established a landmark in the category of algorithms considering a paragraph as a whole rather than proceeding line by line, as earlier (greedy) algorithms used to do [1, 4, 16].

Yet, many aspects of fine typography are not directly or automatically handled by TEX. Consider for example the leftmost paragraph in Figure 1. The typesetting was done by TEX with the Latin Modern Roman font at a 10pt size, and for a paragraph width of 201pt (the figure is optically scaled down in order to fit on the page).

DocEng '24, August 20–23, 2024, San Jose, CA, USA

ACM ISBN 979-8-4007-1169-5/24/08
https://doi.org/10.1145/3685650.3685666

Notice how three lines near the end of the paragraph end the same way, with the word "and". This is considered a defect in high quality typography, as it generates a micro-interruption: the reader's attention may be caught by the similarity and temporarily loose focus or concentration. Such similarities may also lead the reader to accidentally skip a line or re-read the same one, even more so when the problem occurs at the beginning of the line rather than at the end of it. In fact, the field of textual criticism has identified the problem and its consequences in the very ancient context of scribal errors (*e.g.* missing lines in manual copies of the bible made by monks) [20].

We have implemented an extension to the Knuth-Plass (KP) algorithm that is able to deal with that kind of defect. In this paper, we use the term *similarity* for the lack of an official terminology. We would also like to propose the expression *character / word ladder*, analogous to the more widespread expression *hyphenation ladder*, the meaning of which should be obvious. Accidental line skipping has been referred to with a rather awkward French expression, *saut du même au même* ("jump from same to same"), even in non French litterature [20]. Borrowed from rhetoric, the terms *homeoarchy* and *homeoteleuton* have come to designate beginning and end of line similarities respectively, and by extension, accidental line skipping because of them [15].

This paper is organized as follows. Section 2 mentions some related work. Section 3 provides an outline of the KP algorithm's operation, necessary to understand how our extension works. Section 4 describes our extension, and Section 5 presents some experimental results.

2 RELATED WORK

Frank Mittelbach mentions the similarity problem in a survey of existing alternative TEX engines and remaining issues [13]. No solution is proposed in this paper, as it is merely a state-of-the-art review. Alex Holkner addresses the problem in his multiple-objective approach to line breaking [7]. However, the paper only seems to be concerned with beginning-of-line similarities (called *stacks*). Although inspired from it, the approach is not technically an extension to TEX's algorithm. The paper does not provide a precise definition for stacks, and does not use the corresponding objective function in the reported experimental results. Other extensions to the KP algorithm have been proposed in the past, some micro-typographic [17], some macro-typographic, for example to help with automatic document layout [6]. Concerning the latter, our underlying motivations are in fact opposite. The work in question attempts to provide flexibility *at the expense of quality* in order to cope with situations in which manual intervention is impossible, such as automatically adjusting to different displays. We, on the other hand, are interested

In olden times when wishing still helped one, there lived a king whose daughters were all beautiful; and the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. Close by the king's castle lay a great dark forest, and under an old lime-tree in the forest was a well, and when the day was very warm, the king's child went out into the forest and sat down by the side of the cool fountain; and when she was bored she took a golden ball, and threw it up on high and caught it; and this ball was her favorite plaything.

In olden times when wishing still helped one, there lived a king whose daughters were all beautiful; and the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. Close by the king's castle lay a great dark forest, and under an old lime-tree in the forest was a well, and when the day was very warm, the king's child went out into the forest and sat down by the side of the cool fountain; and when she was bored she took a golden ball, and threw it up on high and caught it; and this ball was her favorite plaything.

In olden times when wishing still helped one, there lived a king whose daughters were all beautiful; and the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. Close by the king's castle lay a great dark forest, and under an old lime-tree in the forest was a well, and when the day was very warm, the king's child went out into the forest and sat down by the side of the cool fountain; and when she was bored she took a golden ball, and threw it up on high and caught it; and this ball was her favorite plaything.

Figure 1: Similarity avoidance

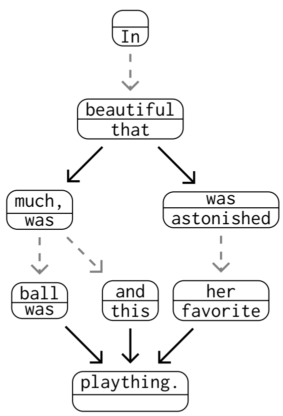

Figure 2: Line breaking graph excerpt

in helping professional (human) typographers aiming for the finest, by providing as much quality as possible automatically, and letting them focus on what strictly requires human intervention. This is more in line with the view of Hurst *et al.* [8].

3 THE KNUTH-PLASS IN A NUTSHELL

The KP algorithm essentially expresses the paragraph justification problem as a *Single Pair Shortest Path* one [3, 5]. The possible solutions for breaking a paragraph into lines are represented in the form of a *graph* (see Figure 2), in which every possible line break is a *node*, and the ability to go from one line break to the next is represented by an *edge* connecting two nodes. In the figure, each node advertises the corresponding end-of-line, and the beginning of the next. The problem thus boils down to finding the best route (referred to as "shortest path" in graph theory) from the top node to the bottom one.

In order to perform efficiently, the KP algorithm uses a *dynamic programming* [2] optimization technique which allows it to never construct the full (potentially huge) graph in memory. As it progresses through the paragraph's text, TEX maintains branches representing the best possible solutions (note the plural) *so far*, but also gets rid of branches which are provably going to be sub-optimal in the end.

Finding the shortest path between two nodes in a graph is usually expressed by minimizing cost rather than maximizing gain. When the KP algorithm explores the line breaking possibilities, it assigns *demerits* to various aspects of the solutions and eventually chooses the cheapest one. TEX's demerits are adjustable parameters, and can be classified in two categories.

Local Demerits are computed line by line, independently of the surrounding context. In particular, the so-called *badness* of a line increases as it needs to be stretched or shrunk, and additional penalties are applied to hyphenated lines.

Contextual Demerits are applied by comparing some aspects of two consecutive lines. In particular, hyphenation ladders are heavily penalized, as well as "adjacency" problems (two consecutive lines with very different scaling; for example a very loose line followed by a very compact one).

The KP algorithm works by minimizing the sum of local and contextual demerits across the whole paragraph.

4 SIMILARITY HANDLING

In order to address similarity problems, our extension works by introducing a new kind of contextual demerits that we call *similar demerits*. Each time the algorithm compares two consecutive lines, it now also compares the respective beginnings and ends of line, and adds the similar demerits value to the total if similarities are encountered. Note that the end of the paragraph needs a special treatment. Most of the time, the last line is not justified, so the potential similarities would not be vertically aligned. Similar demerits may be applied in the very rare cases of complete justification though.

In order to compare consecutive lines for similarities, each node representing a potential break point in the partial graph must remember how the corresponding lines begin and end (which is apparent in Figure 2). This is done as follows. Every time the algorithm creates a new node, it collects the characters from the end of the line backward, discarding kerns, and stopping at the first glue or hyphenation point. The same process is applied (forward) to the beginning of the next line. There are several reasons for doing it this way.

First of all, kerns do not drastically change the vertical alignment of characters, so they have little impact on the reader's perception of similarity (besides, they would be identical in the middle of a similarity). On the other hand, glues, which are elastic, are much more likely to have a noticeable impact on vertical alignment. By stopping before the first one, we thus do not need to compare the

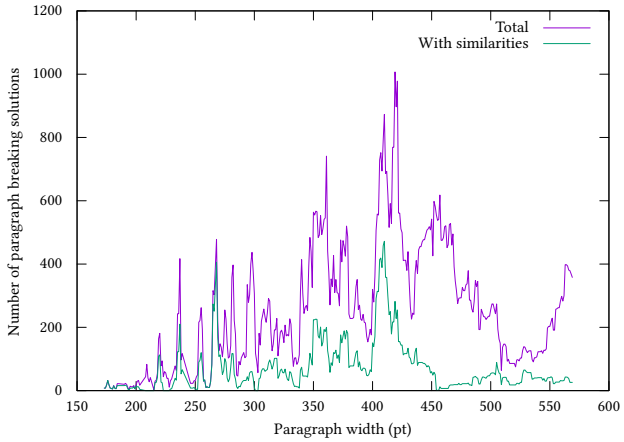

Figure 3: Cross-Width Similarity Report

vertical alignment of the similarities; we just need to compare the sequences of characters, which can be done much more efficiently. Finally, stopping at the first hyphenation point avoids the complexity of deconstructing TEX's discretionaries for content introspection. Discretionaries are slightly more complicated objects used in particular for handling hyphenation and ligatures. Note that by detecting a short similarity (typically one syllable), we implicitly handle a larger one as well, if it exists. In the end, testing for similarity boils down to comparing two short sequences of characters, and finding the longest common sub-sequence.

The first paragraph in figure 1 was obtained with a value of 0 for similar demerits, that is, as TEX itself would do it. With a value slightly above 2800, we obtain the middle paragraph in the figure. The algorithm was able to get rid of the second similarity by pushing the third occurrence of "and" to the beginning of the next line (the one before last), at the expense of more line stretching. This also had the effect of shifting the remainder of the paragraph by two words.

If we push similar demerits to a value slightly above 5230, we obtain the rightmost paragraph in the figure. This time, no more similarity remains, but the algorithm had to change the layout starting as early as at the fourth line to get this result. This paragraph also looks better than the second one in terms of adjacency problems: a close look at the two before last lines in the middle paragraph reveals that they are quite loose compared to the surrounding ones. Finally, a professional typographer to whom we showed the figure finds that the third version globally improves over TEX's original choice, as there is no more hyphenation, and fewer rivers.

5 EXPERIMENTS

In order to assess both the pertinence of the question and the efficacy of our solution, we ran two orthogonal experiments: a single paragraph typeset at many different widths, and many different paragraphs typeset at a single width.

5.1 Pertinence

The purpose of the first experiment is to assess the pertinence of the question. Is the similarity problem a frequent one, and is it worth addressing? In honor of the KP algorithm's founding paper [12], we took an English version of the Grimm Brothers' "Frog King" novel, paragraph 1, and typeset it at all widths ranging from 142pt (approx. 5cm) to 569pt (approx. 20cm) with our own implementation of the KP algorithm. For each of the 427 passes, we recorded the total number of possible layouts, and the number of layouts containing similarities. The results are reported in Figure 3.

Note that because TEX's paragraph breaking algorithm is optimized (see Section 3), it does not normally compute all the possible solutions. On the other hand, our personal implementation comes in two flavours: the regular, dynamically optimized one, and a variant working on the complete solutions graphs. This is how we are able to record the total number of possible layouts.

The figure exhibits a number of interesting characteristics. First of all, the well known inherent instability of the paragraph breaking problem is clearly visible in the chaotic shapes of the curves. Next, and also unsurprisingly, the two curves are very close to each other for smaller paragraph widths. This illustrates the fact that narrow paragraphs are notoriously difficult (sometimes impossible) to justify, and that similarities may be unavoidable. On the other hand, the figure also indicates that in the vast majority of the cases, there is a lot of similarity-free layouts to choose from. All in all, this experiment proves that the similarity problem is indeed a frequent one, but also that getting rid of similarities is an achievable goal.

The next logical question is thus the following: given that in most cases, there exist many similarity-free layouts, will TEX choose one of those, or will it favor a layout with similarities instead? Further analysis of the results gives us the following answers. In 4% of the cases, it is impossible to get rid of similarities (meaning that all possible solutions contain some). Otherwise, when there is a choice, TEX favours a layout with similarities in 21% of the cases. A similar calculation for the second experiment (see Section 5.2) puts this figure at 26%.

21% (26% respectively) is far from being negligible. In very concrete terms, it means that a professional typographer aiming at high quality typesetting would have to manually intervene on two paragraphs out of ten, in order to decide whether they can be improved or not. As a matter of fact, this very paper contains at least a dozen similarity problems, including three very bad ones… In our view, this justifies the claim that the similarity problem is not only frequent, but also worth addressing.

5.2 Efficacy

For the second experiment, we took the text of Herman Melville's Moby Dick novel, freely available from Project Gutenberg's website[1]. We "cleaned up" the text by removing artefacts such as table of contents, chapter names, and in general, all pieces of text that would lead to paragraphs of less than two lines. The resulting corpus contains 1524 paragraphs that we typeset at 284pt (approx. 10cm).

Combined with the first experiment's 427 runs, this amounts to a total of 1951 individual cases, which we ran in three different experimental conditions (for a total of 5853 individual passes) as

[1] https://www.gutenberg.org/

follows. The first batch was typeset with TEX's default settings, thus, not handling similarities at all. The second batch was typeset by maximizing the cost of similarity (similar demerits set to 10 000). Finally, the third batch was typeset by not only penalizing similarities, but also disregarding adjacency problems (adjacent demerits set to 0; see Section 3).

When we set the similar demerits to the maximum value, 48 (first experiment) to 50% (second one) of the problematic paragraphs are "corrected", in the sense that no similarities remain. In the cases where a paragraph contains multiple similarities (up to four in the Moby Dick experiment), the algorithm can sometimes only reduce their number. In such a situation, the number of "improved" paragraphs increases to 50 and 63% respectively. If we not only penalize similarity, but also disregard adjacency problems, 53 to 66% of the problematic paragraphs are corrected, and a total of 57 to 73% are globally improved. These figures clearly demonstrate that similarity can be treated in an automated fashion up to a notable proportion.

Note that completely discarding adjacent demerits is nonsensical from an aesthetic point of view. More generally, just because the algorithm finds a similarity-free layout does *not* mean that it will necessarily look better to a professional typographer. Further experimentation is planned to address that (see Section 7). The purpose of these two experiments is rather to evaluate the leeway we have in similarity handling by studying extreme conditions, and again, the figures clearly demonstrate that the problem can be addressed in the vast majority of the cases.

6 CONCLUSION

We have proposed an extension to the KP algorithm that is able to address similarity problems in paragraph justification. This extension is implemented in our open source platform[2] for typesetting experimentation and demonstration [18, 19] and could be incorporated into any TEX based or inspired system (alternative *TeX engines, Boxes and Glue[3], Typeset[4], InDesign [9], *etc.*). This extension is both simple and lightweight, so it is expected to have a negligible impact on performance, should it be used in production. In fact, a recent conversation with two people involved in LuaTEX confirms that paragraph breaking is *not* a performance bottleneck today. It is also worth noting that this extension is backward-compatible with TEX, in the sense that setting the similar demerits to 0 effectively deactivates it.

7 PERSPECTIVES

Experimentation has demonstrated that treating similarity automatically is both a worthy and achievable goal; Figure 1 illustrates that. On the other hand, getting rid of similarities implies a necessary trade off with other aesthetic criteria [7, 14], for a result the quality of which is ultimately in the eye of the typographer. In the near future, we intend to work hand in hand with professional typographers in order to find a suitable default value for our similar demerits, and also to figure out the acceptable trade offs with the other adjustable penalties and demerits in TEX.

Another area of further experimentation is to *not* limit ourselves to a constant (albeit adjustable) amount for similar demerits. We could for example weight homeoarchies and homeoteleutons differently, penalize similarities proportionally to their length, or even increase the cost of consecutive similarities in a non-linear fashion, so as to penalize ladders more heavily. As a matter of fact, this idea is also applicable to TeX's original demerits (adjacent and double-hyphen ones in particular), and we are already investigating in that direction.

ACKNOWLEDGMENTS

The author would like to thank Thomas Savary, Hans Hagen, and Mikael Sundqvist for some fruitful exchanges.

REFERENCES

[1] Michael P. Barnett. *Computer Typesetting: Experiments and Prospects.* M.I.T. Press, Cambridge, Massachusetts, USA, 1965.
[2] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–516, 1954.
[3] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959. ISSN 0029-599X. doi: 10.1007/BF01386390. URL https://doi.org/10.1007/BF01386390.
[4] Paul E.Justus. There is more to typesetting than setting type. *IEEE Transactions on Professional Communication*, PC-15(1):13–16, 1972. doi: 10.1109/TPC.1972.6591969.
[5] Peter Hart, Nils Nilsson, and Bertram Raphae. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
[6] Tamir Hassan and Andrew Hunter. Knuth-Plass revisited: Flexible line-breaking for automatic document layout. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, DocEng'15, page 17–20, Lausanne, Switzerland, 2015. Association for Computing Machinery. ISBN 9781450333078. doi: 10.1145/2682571.2797091.
[7] Alex Holkner. *Global Multiple Objective Line Breaking.* PhD thesis, School of Computer Science and Information Technology, RMIT University, Melbourne, Australia, October 2006.
[8] Nathan Hurst, Wilmot Li, and Kim Marriott. Review of automatic document formatting. In *Proceedings of the 2009 ACM Symposium on Document Engineering*, DocEng'09, page 99–108, Munich, Germany, 2009. Association for Computing Machinery. ISBN 9781605585758. doi: 10.1145/1600193.1600217.
[9] Eric A. Kenninga. Optimal line break determination. US Patent 6,510,441, January 2003.
[10] Donald E. Knuth. *The TEXbook*, volume A of *Computers and Typesetting.* Addison-Wesley, MA, USA, 1986. ISBN 0201134470.
[11] Donald E. Knuth. *TEX: the Program*, volume B of *Computers and Typesetting.* Addison-Wesley, MA, USA, 1986. ISBN 0201134373.
[12] Donald E. Knuth and Michael F. Plass. Breaking paragraphs into lines. *Software: Practice and Experience*, 11(11):1119–1184, 1981. doi: 10.1002/spe.4380111102.
[13] Frank Mittelbach. E-TEX: Guidelines for future TEX extensions – revisited. *TUGboat*, 34(1), 2013.
[14] Peter Moulder and Kim Marriott. Learning how to trade off aesthetic criteria in layout. In *Proceedings of the 2012 ACM Symposium on Document Engineering*, DocEng'12, page 33–36, Paris, France, 2012. Association for Computing Machinery. ISBN 9781450311168. doi: 10.1145/2361354.2361361.
[15] Stephen R. Reimer. Manuscript studies: Medieval and erly modern. https://sites.ualberta.ca/~sreimer/ms-course/course/scbl-err.htm, 1998.
[16] R. P. Rich and A. G. Stone. Method for hyphenating at the end of a printed line. *Communications of the ACM*, 8(7):444–445, July 1965. ISSN 00010782. doi: 10.1145/364995.365002.
[17] H. T. Thành. Micro-typographic extensions to the TEX typesetting system. *TUGboat*, 21(4), 2000.
[18] Didier Verna. ETAP: Experimental typesetting algorithms platform. In *15th European Lisp Symposium*, pages 48–52, Porto, Portugal, March 2022. ISBN 9782955747469. doi: 10.5281/zenodo.6334248.
[19] Didier Verna. Interactive and real-time typesetting for demonstration and experimentation: ETAP. In Barbara Beeton and Karl Berry, editors, *TUGboat*, volume 44, pages 242–248. TEX Users Group, TEX Users Group, September 2023. doi: 10.47397/tb/44-2/tb137verna-realtime.
[20] Martin Litchfield West. *Textual Criticism and Editorial Technique.* B. G.Teubner, Stuttgart, 1973.

[2] https://github.com/didierverna/etap
[3] https://boxesandglue.dev/
[4] https://github.com/bramstein/typeset/