Towards More Homogeneous Paragraphs

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

Abstract

Paragraph justification is based primarily on shrinking or stretching the interword blanks. While the blanks on a line are all scaled by the same amout, the amount in question varies from line to line. The quality of a paragraph's typographic color largely depends on the aforementioned variation being as small as possible. Yet, TEX's paragraph justification algorithm addresses this problem in a rather coarse fashion. In this paper, we propose a refinement to the algorithm allowing to improve the situation without disturbing the general behavior of the algorithm too much, and without the need for manual intervention. We analyze the impact of our refinement on a large number of experiments through several statistical estimators. We also exhibit a number of typographical traits related to whitespace distribution that we believe may contribute to our perception of homogeneousness.

CCS Concepts

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

Keywords

Paragraph Justification, Paragraph Spacing, Fitness Classes, Adjacent Demerits, TEX, Knuth-Plass

ACM Reference Format:
Didier Verna. 2025. Towards More Homogeneous Paragraphs. In *ACM Symposium on Document Engineering 2025 (DocEng '25), September 2–5, 2025, Nottingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3704268.3742696

1 Introduction

Consider the paragraph in Figure 1(a), which is typeset with a Latin Modern Roman font at a 10pt size, and for a paragraph width of 201pt. This particular layout is the breaking solution chosen by TEX [7, 8], Donald Knuth's famous typesetting system. In spite of the notoriously high quality of its paragraph justification algorithm, the so-called *Knuth-Plass (KP)* [9], this paragraph suffers from a number of deficiencies. In [18], we addressed the "similarity problem" (lines beginning or ending with the same sequence of characters or words), while noticing that the paragraph in question was also defective in terms of whitespace distribution.

Whitespace distribution deals with the amount of scaling (stretching or shrinking) of the interword blanks across the whole paragraph. Whichever algorithm is in use, and absent micro-typographic adjustments [15], scaling interword blanks always is the primary way to adjust the width of a line for justification. Among other things, the overall quality of the typographic color depends on the homogeneousness of the spacing across the whole paragraph. A close look at Figure 1(a) reveals that TEX's breaking solution is sub-optimal in that regard: the beginning of the paragraph is typeset in a rather loose fashion, whereas the end of the paragraph is rather tight. This makes the paragraph look somewhat shaky, and in general, too much spacing discrepancy between consecutive lines may even disrupt the reading experience.

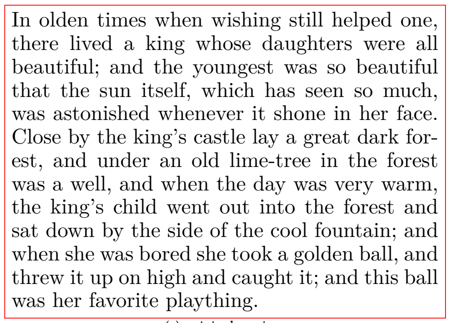
While it is not surprising to encounter similarities in paragraphs typeset by the KP (the original algorithm has no notion of it, so it cannot address it), witnessing homogeneousness problems is more surprising for two reasons. First of all, the KP is not a greedy algorithm as some other algorithms proposed in earlier days [1, 3, 13]. Greedy algorithms operate only locally, line after line, so they cannot naturally modulate their decisions with cross-paragraph considerations. The KP, on the other hand, was one of the very first algorithms to defer the final decision until the whole paragraph was processed. The second reason is that the KP is in fact already aware of such homogeneousness problems, and is even equipped to fight against it. More specifically, consecutive lines with too much spacing variation are heavily penalized, leading to favor more homogeneous solutions. It thus appears that, at least in the case of Figure 1(a), the countermeasures in place fail to operate efficiently.

In this paper we analyze the reason for this failure and we propose a refinement allowing the algorithm to perform better without departing too much from its original behavior. This paper is organized as follows. Section 2 mentions some related work. Section 3 explains how the KP handles the distribution of blanks and analyzes its limitations. Section 4 presents a refinement to the original algorithm, both from a theoretical and practical point of view. Section 5 analyzes the direct impact of this refinement on the outlined deficiencies of the original algorithm. Sections 6 and 7 push the analysis further, notably studying the indirect impact of our refinement on typographical traits that we think may contribute to our perception of homogeneousness, or that contribute to the quality of the typographic color in general. Section 8 gives a word about performance considerations. Finally, sections 9 and 10 conclude and provide some perspectives for future work.

2 Related Work

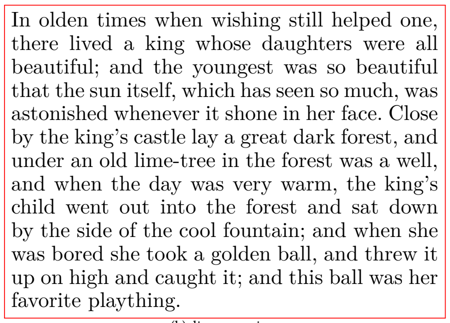
Homogeneousness considerations are evoked by Frank Mittelbach in a survey of existing alternative TEX engines and remaining issues [11]. The analysis is that the (discrete) number of "types of line quality" (see Section 3) is too coarse to perform efficiently. Our

DocEng '25, Nottingham, United Kingdom
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1351-4/2025/09
https://doi.org/10.1145/3704268.3742696



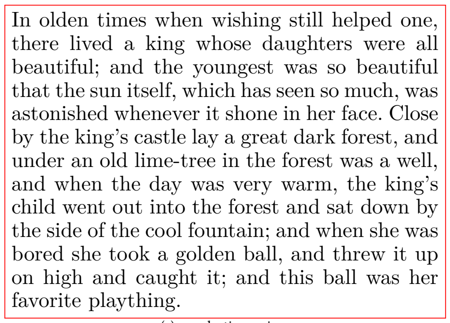
In olden times when wishing still helped one,
there lived a king whose daughters were all
beautiful; and the youngest was so beautiful
that the sun itself, which has seen so much,
was astonished whenever it shone in her face.
Close by the king's castle lay a great dark for-
est, and under an old lime-tree 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.

(a) original version



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.

(b) linear version



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.

(c) quadratic version

Figure 1: Homogeneousness

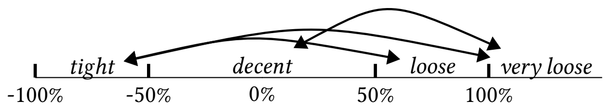


Figure 2: Fitness Classes

analysis goes further along the same lines, and we also propose a working solution in this paper.

Alex Holkner proposes to minimize the variance of the interword spacing as one specific objective in a multiple-objective approach to line breaking [4]. One surprising aspect of his work is that the author seems to favor tight spacing (see his definition of *µLooseness*), whereas the KP algorithm favors natural spacing ("natural" being defined by the font designer). Our approach to the problem is different in that we are trying to refine the KP algorithm rather than completely depart from it. Also, rather than trying to minimize a variance of some sort as an explicit *a priori* objective, we do use a similar measure, only *a posteriori*, as a means to assess the effectiveness of our solution.

In a private conversation, Hans Hagen (the author of LuaTEX[1]) mentions having tried to increase the aforementioned number of "types of line quality", without getting very convincing results. While we also propose something similar, we do believe that the important question is not so much about the number of such types of line quality, as about what exactly to do with them. Our proposition differs from what was attempted in LuaTEX, and our results are statistically convincing.

3 Fitness Classes and Adjacent Demerits

In order to penalize too much discrepancy in the distribution of blanks across a whole paragraph, the KP implements a simple mechanism that boils down to comparing two consecutive lines ("adjacent" in the KP terminology) with each other.

The interword space is normally defined by three values: its normal width, and the amounts of acceptable shrinking and stretching. Based on this, each line has a natural width, and total amounts of acceptable shrinking and stretching respectively. In the remainder of this paper, the proportion of acceptable shrinking (negative) or stretching (positive) that is used for justification is called the Line Spacing Adjustment Ratio (LSAR). A LSAR of 0 means that the line is at its natural width.

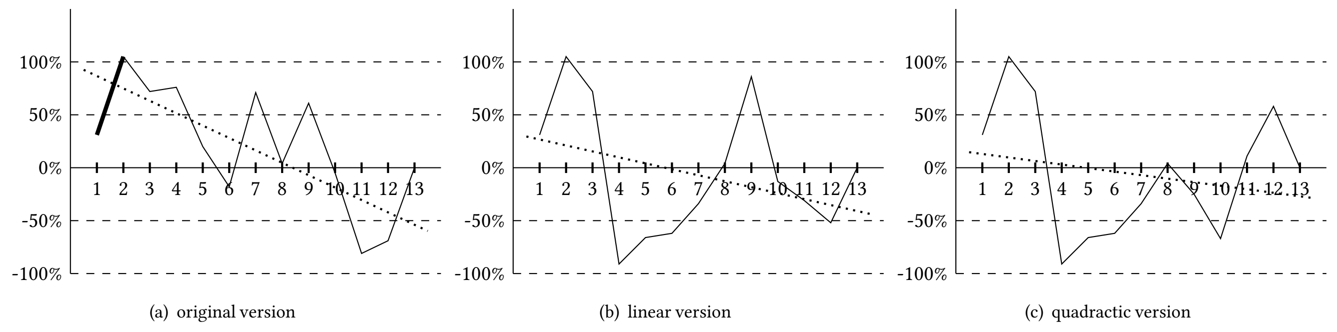
Depending on this ratio, the KP defines four so-called *fitness classes*, as depicted in Figure 2. Decent lines use at most half the acceptable amount of shrinking or stretching, tight lines are shrunk by more than half of the maximum allowed shrinking, loose lines extend between half and 100% of the allowed stretching, and very loose lines are stretched beyond 100%. Note that the KP never uses more than the maximum recommended shrinking, as it would become difficult to distinguish one word from the next below that threshold. That explains the asymmetry in the distribution of fitness classes.

When the algorithm compares two consecutive lines, it penalizes a bad choice by adding *demerits* when the respective fitness classes of the lines are more than one class apart. The demerits in question are adjustable through the `\adjacentdemerits` parameter (`\ad` for short in the remainder of this paper), the value of which is 10000 by default. Thus, adjacent demerits are applied in the three different situations below:

(1) tight ⟷ loose,
(2) tight ⟷ very loose,
(3) decent ⟷ very loose.

In this context, the problem with the paragraph in Figure 1(a) becomes apparent when we look at the LSAR of each successive line. This is depicted in Figure 3(a), where it is indeed visible that the paragraph is rather loose at the beginning, and rather tight towards the end. Each of the 13 lines appears on the horizontal axis, with its LSAR (expressed as a percentage) on the vertical axis. The exact values are not important. What matters here is to notice that adjacent demerits are applied *only once* on this whole paragraph, namely between lines 1 and 2. This is because line 1 is decent whereas line 2 is very loose, although only by a very small amount (in fact, just 4% above the 100% threshold). Every other pair of consecutive lines are considered adjacent (including for example lines 10 and 11 which happen to be further apart from each other than lines 1 and 2), so no more adjacent demerits are applied. The conclusion is then obvious: fitness classes work in such a way that most of the adjacency problems in this paragraph are simply ignored.

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



(a) original version (b) linear version (c) quadractic version

Figure 3: Line Spacing Adjustment Ratios

4 Fitness Classes Refinement

Rather than suggesting a completely different approach, what we propose in this paper is to refine the way fitness classes are defined and used, mostly for the potential applicability to production engines using the original version of the algorithm.

The reportedly unconvincing trial attempted in LuaTEX was to increase the number of fitness classes, while still applying adjacent demerits for consecutive lines more than one class apart from each other. It is possible that this approach did not produce better results because it disturbed the algorithm's behavior too much, for example by applying adjacent demerits too quickly or too often. Indeed, one needs to remember that the KP is a complex machinery involving many adjustable parameters affecting aesthetic choices that are, in fact, intertwined. As a result, modulating one particular aspect of the typesetting will inevitably affect the others, in a possibly bad way.

The other important point to consider is that the various formulas used in the KP, along with the default values for its parametrization, are the result of a very careful study conducted by Donald Knuth in the 80's, on large corpuses of text, which eventually were reported to "work well in practice" [9]. This is another incentive to not disturb the workings of the algorithm too much.

4.1 Gradual Demerits

Based on these considerations, the following proposition can be formulated. In its current state, for two consecutive lines in the tight – decent – loose range, the KP will brutally apply adjacent demerits if the LSARs are at least 100% apart (typically, two lines at the two extremes of the decent class). Under that threshold, adjacent demerits remain inoperative. We thus propose to refine this "no man's land" by gradually applying demerits in a way proportional to the LSAR variation, until the actual value of `\ad` is reached for a 100% discrepancy, and at which point the value will remain clamped, as in the original version. Note that if one of the lines is tight, the actual adjacent demerits trigger is at 150%, not 100%. In our refined version, we prefer to keep the 100% threshold across the whole tight – loose range for reasons of symmetry around the ideal LSAR value of 0 (very loose lines are really undesirable). In a similar vein, if one of the two lines being compared is very loose, adjacent demerits are applied as soon as the LSARs are 50% apart, so we also apply gradual demerits before that, only twice as fast.

4.2 Rationale

The idea of gradual demerits is in fact quite natural for two reasons. First of all, it will make the algorithm explore alternative layouts while paying attention to adjacency considerations where they would otherwise have been ignored. Next, the corresponding typographical trait (the difference in the scaling of two consecutive lines) is in fact a continuous one, although the `\ad` trigger is Boolean. This is actually a singularity in the KP. The other existing demerits in TEX do correspond to Boolean typographical traits (for example, `\finalhyphendemerits` corresponds to whether the penultimate line is hyphenated). The only other continuous trait that TEX observes is the LSAR itself, and the corresponding cost is given by the line's so-called *badness*, which is also a continuous function. Thus, gradual demerits are simply a way to acknowledge the continuous nature of the corresponding typographical trait, and to take it into account.

Perhaps the rationale behind clamping deserves more explanation though. Apart from the aforementioned badness of a line, which is computed via a hard-wired formula, all kinds of penalties or demerits in TEX are accessible to the user via adjustable parameters. `\ad` is one of those. If we did not clamp the computed gradual demerits to TEX's original value, we would take the risk of essentially disregarding every other typographical defect known to TEX in the case of consecutive lines with very different scaling. For example, TEX is able to avoid hyphenation ladders by applying `\doublehyphendemerits` when two consecutive lines are hyphenated. By default, the values for this parameter and `\ad` are the same, which means that what works well for TEX is to consider LSAR discrepancies of 100 / 50% *and beyond* as bad as consecutively hyphenated lines. We do not want to rupture that equilibrium, and in fact, avoiding hyphenation ladders is probably preferable to avoiding a very loose line following a tight one.

4.3 Formal Definition

In their simplest form, gradual demerits can be defined as a linear function of the LSAR variation. However, we also wanted to see how this refinement would behave when being more lenient on small variations, and less tolerant on larger ones. Thus, we also tried a quadratic formula. Gradual demerits may hence be formally defined as follows.

Let l_i and $l_{j=i+1}$ be two consecutive lines. Let $r_i = \mathrm{LSAR}(l_i)$ and $r_j = \mathrm{LSAR}(l_j)$ ($r_i, r_j \in [-1, +\infty[$). The demerits d_{ij} applied when comparing l_i and l_j are then calculated like this.

(1) If $r_i, r_j \leq 1$,
$$d_{ij} = \min\left(\texttt{\textbackslash ad}, \texttt{\textbackslash ad} \cdot |r_i - r_j|\right)$$
in the linear case, or
$$d_{ij} = \min\left(\texttt{\textbackslash ad}, \texttt{\textbackslash ad} \cdot |r_i - r_j|^2\right)$$
in the quadratic case.

(2) If r_i or $r_j > 1$,
$$d_{ij} = \min\left(\texttt{\textbackslash ad}, 2\texttt{\textbackslash ad} \cdot |r_i - r_j|\right)$$
in the linear case, or
$$d_{ij} = \min\left(\texttt{\textbackslash ad}, 4\texttt{\textbackslash ad} \cdot |r_i - r_j|^2\right)$$
in the quadratic case.

4.4 Practical Definition

The above formulas are unfortunately incompatible with the dynamic programming optimization technique [2] used in the KP, and this is actually the reason why the corresponding continuous typographical trait was discretized through a fixed number of fitness classes in the first place. Working with continuous functions is only possible if one keeps the full graph of possibilities in memory, which is unrealistic in practice, due to the exponential complexity of the problem.

The necessity for a discrete number of fitness classes may be explained informally as follows. The KP avoids maintaining an exhaustive list of paragraph breaking solutions by keeping track of a limited set of potentially optimal ones only. Suppose that the algorithm has reached line l, at which point it would favor a break making the line tight. Although such an arrangement may look optimal at that point, the algorithm has not processed the next line yet, so it doesn't know its possible kind(s) of fitness. If later on, it turns out that the only possibility for the next line is to make it very loose, such a layout would be heavily penalized, and perhaps it would in fact have been a better choice in the end to make line l loose instead of tight. In other words, a locally optimal choice may turn out to be a bad one after all, so a definitive decision must not be made too early.

In order to avoid making such premature decisions, the KP does not only remember which break points it has found, but *how* it has found them as well. More specifically, for every possible break point b found, the algorithm may remember up to four different ways to reach it; one (and *only* one) for each fitness class[2]. That way, it is able to choose only at the very end which path minimizes the total amount of adjacent demerits.

If, on the other hand, we were to remember the exact LSAR of each possible line ending at b instead of its fitness class, every such ratio would be different for each possible line, which means that we would end up keeping the exhaustive list of solutions in memory until the end. Again, this is unrealistic in practice, due to the exponential complexity of the problem.

4.5 More Fitness Classes

In order to remain compatible with the original dynamic programming optimization, we thus need to sample our continuous gradual demerits into a discrete number of intervals. We choose to create 10%

[2]The case of non rectangular paragraphs introduces even more complication, but it is out of the scope of this paper.

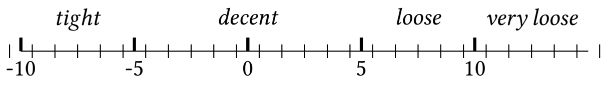


Figure 4: New Fitness Classes

wide centered slices and define one integral fitness class for every such slice. Figure 4 depicts this with a recollection of the original four ones. A line shrunk by 100% belongs to class -10, a line half shrunk to class -5, a line stretched to 100% to class 10, *etc.* Centering the classes allows to keep the symmetry between shrinking and stretching. For example, class 0, our new and very narrow "decent" class extends between LSARs of -5% and +5%. Thus, in this new implementation, the number of fitness classes covering the range tight – loose amounts to 21, as opposed to the 3 original ones. Note that for very loose lines, the number of fitness classes is theoretically infinite, but this is not a problem for two reasons. First of all, we do not need to *predefine* an infinite number of fitness classes, as the class identifiers are integrals computed from the LSAR itself (see below). Second, recall again that in the event of a preposterous stretching, the computed gradual demerits are clamped to the finite value of \ad anyway.

In this context, our refinement can be reformulated as follows. For line l_i with a LSAR of r_i, the fitness class of l_i is defined by $c_i = floor(10r_i + 1/2)$. Now, the demerits d_{ij} applied when comparing l_i and l_j are calculated as follows.

(1) If $c_i, c_j \leq 10$,
$$d_{ij} = \min\left(\texttt{\textbackslash ad}, \frac{\texttt{\textbackslash ad} \cdot |c_i - c_j|}{10}\right)$$
in the linear case, or
$$d_{ij} = \min\left(\texttt{\textbackslash ad}, \frac{\texttt{\textbackslash ad} \cdot |c_i - c_j|^2}{100}\right)$$
in the quadratic case.

(2) If c_i or $c_j > 10$,
$$d_{ij} = \min\left(\texttt{\textbackslash ad}, \frac{\texttt{\textbackslash ad} \cdot |c_i - c_j|}{5}\right)$$
in the linear case, or
$$d_{ij} = \min\left(\texttt{\textbackslash ad}, \frac{\texttt{\textbackslash ad} \cdot |c_i - c_j|^2}{25}\right)$$
in the quadratic case.

5 Experimentation

The paragraphs in Figures 1(b) and 1(c) are the ones obtained when we apply linear and quadratic demerits respectively. Notice how they indeed look more homogeneous, and as a side-effect, are also rid of similarities. As a matter of fact, the solution in Figure 1(c) is the one we obtained in [18] when we addressed the similarity problem explicitly. These new versions share the first three lines with the original one (in fact, there happens to be no other choice there), have the next five lines in common, and subsequently differ on the next four.

This paragraph is but one example of alternative renditions however, and we now need to explore how well our refinement performs at large. In order to answer that question, we set up a number of number of experiments based on corpuses of text we had prepared

	Frog King	Moby Dick
All algorithms agree	43%	64%
Linear > quadratic	35%	46%
Quadratic > linear	65%	54%
Linear ∈ KP's top 10%	78%	63%
Quadratic ∈ KP's top 10%	81%	62%

Table 1: KP Disruption

for [18]: the first paragraph of the Grimm Brothers' "Frog King" novel appearing in Figure 1, typeset at 379 different widths, and 1279 paragraphs from the "Moby Dick" novel, typeset at the single width of 284pt (approximately 10cm). This amounts to a total of 1658 individual experiments.

5.1 KP Disruption

Given that one of our initial concerns was to not disturb the original algorithm too much, we are interested in knowing how far the alternative layouts selected by our refinement are from the original KP algorithm's choice. In order to answer that question, we recorded, for each of the 1658 individual experiments, *all* the possible solutions found by an unoptimized implementation of the original KP algorithm, and sorted them by decreasing estimated quality (*i.e.*, decreasing total demerits in TEX's jargon). We then looked up the rank of our linear and quadratic layouts in that sorted and exhaustive list of solutions. A rank of 0 indicates that an alternative choice is the same as the original one. Table 1 summarizes the results.

Depending on the exact scenario (Frog King or Moby Dick), the three algorithms offer the same solution in 43 and 64% of the cases respectively. Put the other way around, this means that our refined algorithm is able to choose an alternative solution in 36 to 57% of the cases, which is far from negligible, especially since those alternative solutions are found automatically, without even touching the parametrization of the original KP (the value of `\ad` remains the same; the default one). Note that the actual number of possible layouts for an individual experiment is extremely chaotic. Some paragraphs have only one or very few possibilities, some other may have up to 90000 alternatives. If we filter out the cases where there is only one choice, the numbers do not vary significantly so they are not reported here.

The second interesting result is that whatever the experiment, the quadratic solutions look predominantly better in the eyes of the original KP: they are graded with fewer demerits than the linear ones in 54 to 65% of the cases respectively (> in Table 1 means better, so the ranking is in fact smaller).

Finally, and this is perhaps the most important outcome, in the cases where an alternative layout is chosen, it remains within the KP's top 10% choices in 62 to 63% of the cases for the Moby Dick experiment, and in 78% to 81% of the cases for the Frog King one. If we include the cases where the algorithms agree with each other, the percentages increase to 88 – 90%. In fact, across the full set of experiments, the average rank of the alternative solutions is around 3.5 – 4%. In other words, the alternative layouts chosen by our refined algorithm would practically always be considered as fairly acceptable by the original version. Note however that the "top 10%" estimator

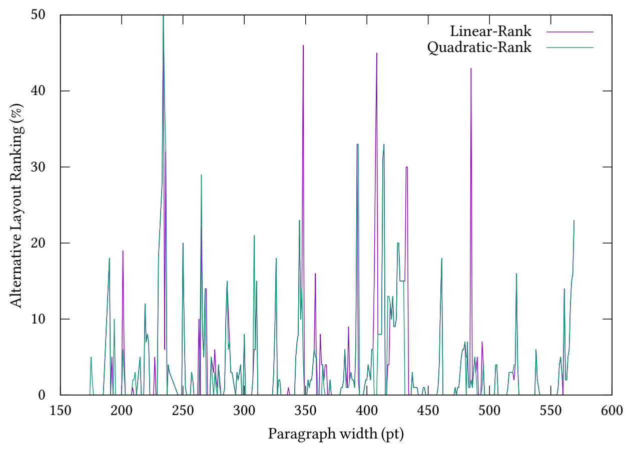


Figure 5: Frog King Alternative Layouts Rankings

is rather arbitrary. As mentioned before, the room for choice is extremely volatile and of course, when the linear or quadratic layout chosen happens to be the second one out of three or four, it doesn't fall in the top 10%.

Figure 5 provides a more detailed view of the rankings (as percentages of the total number of possible solutions) of the linear and quadratic layouts in the eyes of the KP for the Frog King experiment. Again, a ranking of 0 means that the algorithms agree with each other, and a ranking of 100% would be the KP's last choice. Apart from the aforementioned chaotic nature of the problem, this figure makes it visible that *all* alternative layouts remain within the first half choices of the KP (the worst choice is at 50%). A similar figure for the Moby Dick experiment[3] also exhibits a 50% threshold, although with some very rare exceptions for which the ranking would fall down to 70%.

Nevertheless, these statistics tend to confirm that we have indeed succeeded in proposing a refinement that does not disturb the original behavior too much.

5.2 Adjacency

Apart from our concern for disruption, the primary goal of this project was to have the KP pay more attention to adjacency considerations, that is, to the variation of the LSARs of two consecutive lines. We thus need to assert the efficiency of our gradual demerits in reducing that variation.

In order to do that, we adapted our experiments to record the LSARs of every selected choice in the quadratic, linear, and original algorithm versions. But we also need a statistical indication of how well our refinement performs. Considering that in the ideal case, all lines would have the same LSAR (in other words, a difference of 0), we can estimate how distant we are from this optimum by computing the Root Mean Square (RMS) of the differences between consecutive LSARs. In other words, we define the Adjacency Root

[3]Given the number of paragraphs in the Moby Dick experiment, such figures would be too dense to be readable in a PDF, so they are not included in this paper. They are however available as auxiliary material in SVG format.

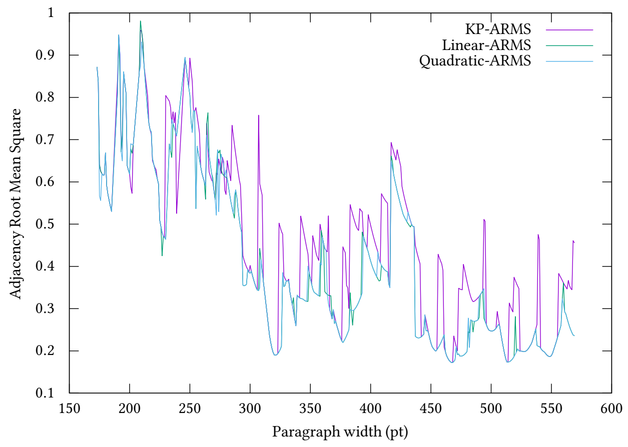


Figure 6: Frog King Adjacency Root Mean Square

	Frog King	Moby Dick
KP average ARMS	0.45	0.56
Linear average ARMS	0.40	0.53
Quadratic average ARMS	0.39	0.53
Linear ARMS < KP ARMS	89%	81%
Quadratic ARMS < KP ARMS	92%	86%

Table 2: Adjacency Root Mean Square

Mean Square (ARMS) of a paragraph as follows.

$$A_{rms} = \sqrt{\frac{\sum_{i=1}^{n-1}(r_i - r_{i+1})^2}{n-1}}$$

Figure 6 provides a plot of the ARMS for the Frog King experiment. The first remark is that whatever the algorithm version, the ARMS has a tendency to decrease as the paragraph grows wider. This is not surprising, considering that narrow paragraphs are notoriously more difficult to typeset (fewer choices), and wide ones have more elasticity available in each line. The equivalent figure for the Moby Dick experiment does not exhibit this behavior, as the paragraph width is fixed.

A more interesting remark is that our refinement performs visibly better than the original KP, sometimes by a large amount, such as around 380pt, or in the 460–500pt range. From the picture itself however, we cannot clearly infer which of the linear or quadratic approach performs better in general. Table 2 provides more concrete results. Compared to the original algorithm, our refinement features a systematically lower average ARMS. When the algorithms provide different layouts, the linear version improves the situation in 81 to 89% of the case, against 86 to 92% for the quadratic version.

From these numbers, we can draw two conclusions. First of all, we see that the quadratic version offers slightly better results than the linear one. The most striking result however is that for nine paragraphs out of ten when there is a choice, we are able to reduce the scaling discrepancies between two consecutive lines. This means not only that there is indeed a lot or room for alternative choices where the original KP simply ignores adjacency problems, but also that our gradual demerits are very efficient at finding them.

	Frog King	Moby Dick
KP average slope	0.05	0.07
Linear average slope	0.06	0.07
Quadratic average slope	0.06	0.07
Linear slope < KP slope	32%	44%
Quadratic slope < KP slope	33%	44%

Table 3: Global Trend Slopes

6 Further Analysis

At that point, we could satisfy ourselves with those results and stop there. There is however more to homogeneousness than meets the eye, and this becomes strikingly apparent when we look at Figures 3(b) and 3(c). These are the LSARs for the linear and quadratic alternative layouts of our sample paragraph respectively.

When compared with the original plot (Figure 3(a)), the reader may now be legitimately surprised by the perceived increase in homogeneousness. Indeed, the LSARs distributions in the two alternative layouts are clearly wider than in the original case. As a matter of fact, the paragraph that serves as an illustration in this paper happens to be a counterexample of the point we are trying to make. In this particular case, the ARMS of the original layout (0.59) turns out to be better than that of the linear one (0.68), itself better than that of the quadratic one (0.69). Thus, it appears that the ARMS is not a sufficient estimator to account for improvement. We formulate two hypotheses that may explain this phenomenon.

First of all, and contrary to the original case, neither the linear nor the quadratic version exhibit a strong tendency towards compaction (or expansion, for that matter) as we progress through the paragraph. Rather, the LSARs seem to oscillate around natural spacing without any noticeable trend. We believe that the absence of such a trend is less disruptive of the paragraph's typographic color as a whole.

The second hypothesis is that even though the LSARs still oscillate a bit, the oscillation *frequency* is lower than in the original case. Consider lines 6 to 10 for example. In Figure 3, we observe a total of four "peaks" ($\wedge$ or $\vee$, *i.e.*, a sudden change in the scaling direction), that is, one at every line but the tenth. In the linear version, there is only one (at line 9), and two (at lines 8 and 10) in the quadratic case. In totality, there are eight peaks in the first paragraph, against four in the linear version, and five in the quadratic one. Also, in spite of the additional quadratic peak, that version happens to be slightly closer to natural spacing in general. It is thus possible that oscillation frequencies impact our perception of homogeneousness.

6.1 Global Trends

One way to detect a global tendency towards expansion or compaction within a single paragraph is to perform a linear regression on the LSARs, and observe the slope of the resulting affine transformation. The dotted lines in Figure 3 represent such linear regressions obtained with the least squares method, and confirm that in the particular case of this paragraph, the tendency to compaction is much more pronounced in the original version than in the linear and quadratic alternatives respectively.

6.1.1 Slopes. Figure 7 presents the linear regression slopes (in absolute value) for the Frog King experiment. One interesting remark

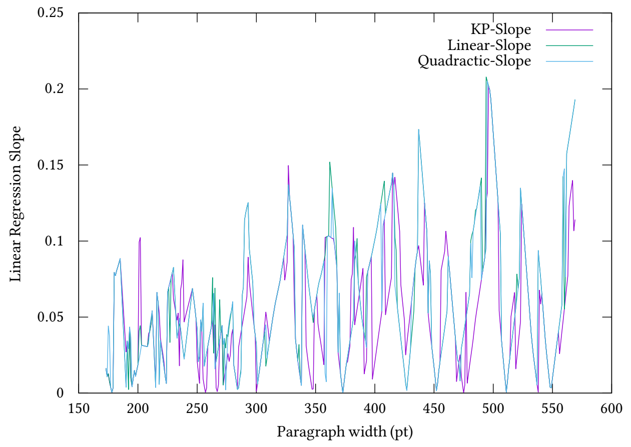


Figure 7: Frog King Global Trend Slopes

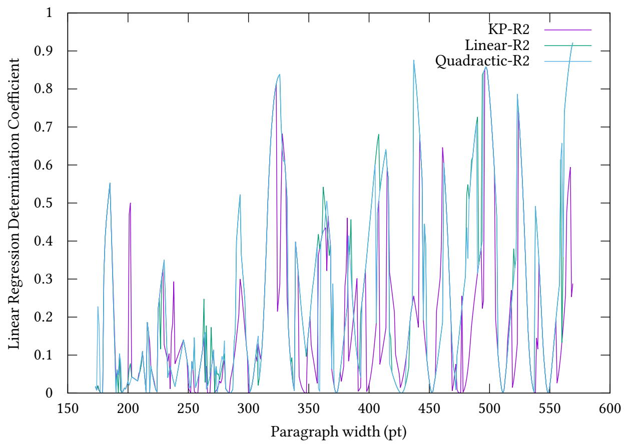


Figure 8: Frog King Global Trend R^2

is that whatever the algorithm, the likelihood of getting steep slopes increases with the paragraph width. Considering that wider paragraphs are easier to typeset, this trait is somewhat surprising and we do not have an explanation for this yet. Apart from that, there is not much that can be clearly deduced from this figure, and further analysis reported in Table 3 reveals that our refinement does not bring any noticeable improvement on the matter. The average slopes are practically the same in all experiments (between 0.05 and 0.07, which is in fact relatively flat). Also, when the alternative layouts are different from the original one, the linear or quadratic versions feature slopes flatter than the original in only 32% to 44% of the cases. It thus appears again that the paragraph in Figure 1 is *not* statistically representative of the general behavior.

There is one final remark to be made on Table 3. In general, the Moby Dick experiment is harsher on the typesetting than the Frog King one, regardless of the algorithm. It also has a tendency to reduce the differences between the original algorithm and our refinement. As an exception to the rule, we can see that it is in fact the opposite when it comes to slopes.

6.1.2 Linear Regression Quality. Linear regressions do not account for every aspect involved in the perception of global trends though. First of all, it could be argued that the number of samples (*i.e.* paragraph lines in our case) available when computing them is too small for linear regressions to be really meaningful all the more for wider paragraphs where the number of lines can fall drastically. Next, the quality of the linear regression itself can vary greatly depending on the experiment. Observe for instance that even though the slopes in Figures 3(b) and 3(c) are flatter than in Figure 3(a), the spread of the samples is actually wider, meaning that the linear regression is *less* representative of the reality.

The quality of the linear regression can be estimated by computing the Determination Coefficient (R^2), a scalar value in $[0, 1]$ (the closer to 1 the better), and confirms the previous remark. In the original case, the R^2 is of 0.47, against 0.06 and 0.02 in the alternative versions. Figure 8 depicts the R^2 for the Frog King experiment. From the general shape of it, it seems that whatever the algorithm, the likelihood of getting linear trends increases with the paragraph width.

	Frog King	Moby Dick
KP average R^2	0.19	0.17
Linear average R^2	0.27	0.18
Quadratic average R^2	0.26	0.19
Linear R^2 > KP R^2	68%	59%
Quadratic R^2 > KP R^2	69%	60%

Table 4: Global Trend R^2

Also, whether the original algorithm or our refinement perform better seems to be highly dependent on the paragraph width. For example, the original KP performs clearly better around a width of 200pt, whereas our refinement takes over at 325pt and 575pt. Table 4 presents some further analysis of the resulting numbers, and this time, our refinement does exhibit interesting improvements. The average R^2 is noticeably better than the original one in the Frog King experiment, although only slightly in the Moby Dick one. More importantly, when the alternative layouts differ from the original one, we also get better R^2 in 68 to 69% of the cases in the Frog King experiment, and 59 to 60% in the Moby Dick one.

To sum things up, it appears that even though our refinement does not help avoiding global trends, it does produce layouts closer to their LSAR linear regressions.

6.2 Peaks

The second hypothesis we formulated is that LSAR oscillation may affect the typographic color. The question here is thus how often do the scalings of consecutive lines change direction, in other words, how many peaks ($\wedge$ or $\vee$) do we observe in plots like those in Figure 3.

Figure 9 shows the number of such peaks for the Frog King experiment. In order to account for the variation in the resulting paragraph height, the number of peaks is expressed as a percentage of the maximum possible number, that is, the paragraph's line number minus two. The figure gives a feeling that our refinement might perform better in this regard (the Moby Dick one would give the same impression), something which is confirmed by further analysis reported in table 5. The average number of peaks is noticeably lower with

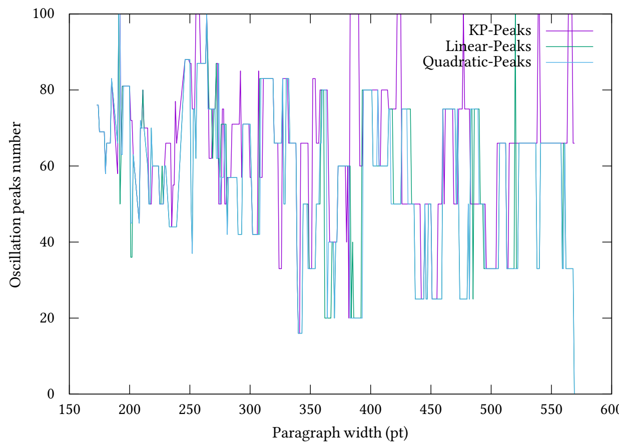


Figure 9: Frog King Oscillation Peaks

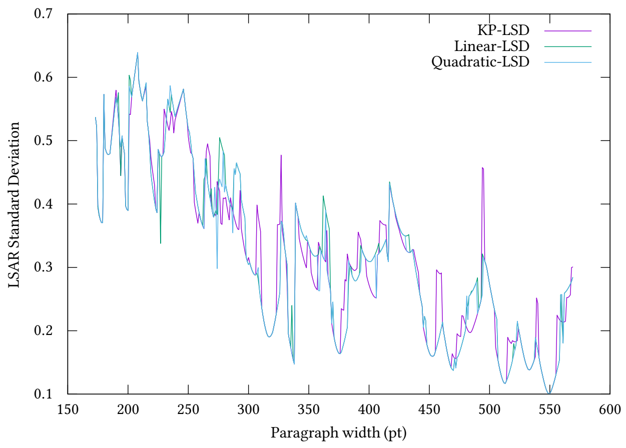


Figure 10: Frog King LSAR Standard Deviation

	Frog King	Moby Dick
KP average peaks	65%	66%
Linear average peaks	56%	62%
Quadratic average peaks	56%	62%
Linear peaks < KP peaks	75%	72%
Quadratic peaks < KP peaks	78%	72%

Table 5: Oscillation Peaks

	Frog King	Moby Dick
KP average LSD	0.31	0.39
Linear average LSD	0.31	0.38
Quadratic average LSD	0.31	0.38
Linear LSD < KP LSD	42%	49%
Quadratic LSD < KP LSD	44%	52%

Table 6: LSAR Standard Deviation

our refinement in the Frog King experiment, less but still better in the Moby Dick one. More importantly, when the alternative layouts differ from the original one, we get fewer peaks in 75 to 78% of the cases for the Frog King experiment, against 72 in the Moby Dick one.

Note that the global improvements in oscillation frequency that our refinement features is all the more beneficial that it is coincidental. Indeed, intentional peak avoidance would require comparing lines three by three, and perhaps penalize them with a new `\peakdemerits` parameter. Instead, our refinement only compares lines two by two, as in the original KP, and does not require any new parameter to improve the situation 72 to 78% of the time.

At that point, the reader may legitimately object that the number of peaks accounts for the oscillation frequency *only*, regardless of its amplitude, and it is very likely that frequent oscillation with a small amplitude would not be so much of a problem. There are different ways to account for amplitude. One of them happens to be the linear regression's R^2 that was discussed in Section 6.1, and the results there were in favor of our refinement. Another possible way to account for amplitude consists in looking at the LSARs standard deviation, in other words, taking the average LSAR as a reference point instead of the linear regression itself. We thus define the LSAR Standard Deviation (LSD) as follows (μ being the average LSAR).

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (r_i - \mu)^2}{n}}$$

Figure 10 provides the LSD for the Frog King experiment. As in the case of the ARMS, there is a global tendency towards smaller amplitudes as the paragraph grows wider. Contrary to the ARMS, there doesn't seem to be a clear winner in general. Table 6 allows us to refine that analysis. The average LSDs are very close to each other, or even identical in the Frog King case. In the Moby Dick experiment, only the quadratic version outperforms the KP, and even so barely. In the Frog King experiment, the situation is slightly more in favor of the original algorithm.

7 Naturalness

As mentioned before, paying more attention to one typographical trait in particular (in our case, adjacency considerations) always comes at the detriment of others [4, 12]. The KP algorithm provides no fewer than ten parameters allowing the user to adjust the relative weight of several such traits. For example, one can give more or less importance to hyphenation by changing the value of `\hyphenpenalty`. One of these traits however is hardwired in TEX: how far a line is from the natural interword spacing (specified by the font designer) is evaluated by the so-called *badness*, which is computed by an internal formula.

Since the user has no direct control over the badness[4], we are interested in knowing how our refinement affects the LSARs with respect, not to each other this time, but to 0. In other words, we want to evaluate the comparative "naturalness" of original, linear, and quadratic layouts. More specifically, we do not expect the badness to globally improve, since we are now paying more attention to

[4]It is in fact possible to request more or less compact paragraphs by adjusting the `\looseness` parameter, but it is only a very limited and somewhat hackish way to do so. It is also possible to override a font's natural spacing specifications, but this is not technically done through some KP parametrization.

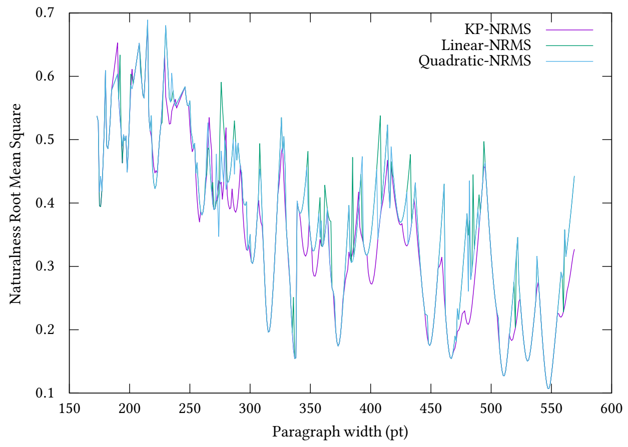


Figure 11: Frog King Naturalness Root Mean Square

	Frog King	Moby Dick
KP average NRMS	0.34	0.43
Linear average NRMS	0.36	0.44
Quadratic average NRMS	0.36	0.44
Linear NRMS < KP NRMS	13%	32%
Quadratic NRMS < KP NRMS	15%	32%

Table 7: Naturalness Root Mean Square

another typographical trait. Rather, we want to know how costly our refinement is in terms of naturalness.

We can estimate how distant we are from natural spacing by computing the root mean square of the LSARs of each line. In other words, we define the Naturalness Root Mean Square (NRMS) of a paragraph as follows.

$$N_{rms} = \sqrt{\frac{\sum_{i=1}^{n} r_i^2}{n}}$$

Figure 11 exhibits the NRMS for the Frog King experiment. We can see that whatever the algorithm, the naturalness has a general tendency to improve as the paragraph grows wider. Again, this is expected as there is notoriously more latitude in typesetting wide paragraphs. Also, the three variants seem to remain quite close to each other, with the original KP looking occasionally better.

Table 7 provides more specific numbers. First of all, the average NRMSs are indeed quite close to each other: our refinement induces a degradation of only 0.01 to 0.02. Next, and as expected, the NRMS remains largely better in the original layout, typically around 85% of the time for the Frog King experiment, and 70% of the time in the Moby Dick one. Also, with respect to the comparative harshness of the experiments, we are faced with a second exception here (the first one being about slopes in Table 3). Contrary to the usual behavior, the Moby Dick experiments allows our refinement to perform better than the Frog King one on the NRMS.

Of course, the cost of our refinement on the NRMS should not be considered in isolation (as shouldn't any of the other statistical estimators used in this paper). In the case of our sample paragraph for example, the NRMS of the original layout is of 0.58 whereas the linear one is at 0.60 and the quadratic one at 0.59. The degradations are admittedly rather small, but nevertheless, the refined layouts are also admittedly better, in spite of the degradation.

8 Performance

Our refinement has been implemented in ETAP[5], an open source platform for typesetting experimentation and demonstration [16, 17]. Rather than being optimized for performance, ETAP is heavily instrumented for experimentation, so it is impossible to accurately measure the impact of our refinement. However, should this refinement be incorporated into production engines, we believe that impact to be minimal for two reasons.

First of all, according to the author of LuaTEX, paragraph justification is not a performance bottleneck anymore, I/O and font processing having taken over. It is thus very unlikely that the additional arithmetic involved in computing gradual demerits would have a noticeable impact.

Next, n being the number of potential break points in a paragraph, remember that the dynamic programming optimization that TEX uses reduces the original (theoretical) complexity of the problem from $O(2^n)$ to a worst-case $O(n^2)$, but in fact often closer to linear time, independent of the paragraph's width [9]. The heart of the algorithm consists in traversing the so-called *active nodes list*, representing the break points currently being tested. The length of this list is of the same order of magnitude as the average number of words per line, so it is quite small. Given the increased number of fitness classes our refinement requires, our new active nodes list can be expected to be around ten times longer than the original one, in the worst case scenario, and for an algorithm that was designed to work in the seventies. This should definitely not be a problem by today's hardware standards.

9 Conclusion

In this paper, we have proposed a refinement to the KP algorithm allowing it to select alternative layouts when scaling discrepancies between consecutive lines would otherwise have been disregarded. This refinement is internal only: it neither modifies, nor extends the user-level interface. Since the only thing our refinement does is to modify the way adjacent demerits are computed, it can easily be incorporated into any TEX-based or inspired system (alternative *TeX engines, Boxes and Glue[6], Typeset[7], InDesign [6], *etc.*), with a performance cost that is expected to be unimpactful.

Section 5 has demonstrated that gradual adjacent demerits are indeed successful in fulfilling both of our original objectives. In the vast majority of the 1658 experiments that were conducted, we are able to select alternative layouts that reduce the scaling discrepancies between consecutive lines, and at the same time that would still be considered as fairly acceptable choices in the eyes of the original algorithm.

However, further analysis of the LSARs in our sample paragraph has demonstrated that our perception of homogeneousness cannot be reduced to adjacency considerations only. We have formulated

[5]https://github.com/didierverna/etap
[6]https://boxesandglue.dev/
[7]https://github.com/bramstein/typeset/

Trait	Impact
Disruption	✓
ARMS (adjacency)	✓
Slope (global trends)	✗
R^2 (linearity)	✓
Peaks (oscillation)	✓
LSD (spread)	–
NRMS (naturalness)	✗

Table 8: Gradual Demerits Impact Summary

a number of hypotheses about other typographical traits that may contribute to it, and we have studied the impact of our refinement on those traits. Our findings are briefly summarized in Table 8, and make us believe that gradual demerits are indeed a useful addition to the KP. In particular, it is notable that the positive impact of our refinement on linearity and oscillation is collateral. The only aspect that gradual demerits influence directly is the ARMS.

Also, our analysis allows us to conclude that the quadratic version of our refinement performs consistently better than the linear one, although the various statistical estimators only exhibit relatively small differences between the two variants.

Finally, and with a couple of exceptions, the Moby Dick experiment is harsher on the three algorithmic variants than the Frog King one. That experiment should probably be considered more representative of the general case, as the corpus of text is much bigger. Nevertheless, one interesting aspect of the Frog King experiment is to show the influence of the paragraph width on the behavior of the algorithm, and experience shows that our refinement does not introduce any fundamental divergence from the original version.

10 Perspectives

Apart from potential inclusion in production engines, the perspectives of this work are numerous.

First of all, we do not yet know whether there exist any correlations between the typographical traits we have analyzed. If such correlations are to be found, they may turn out to be important. For example, consider that the perception of a strong global trend is likely to be attenuated by a large spread of the LSARs (low R^2 or high LSD), whereas a soft but highly linear slope will probably be more visible. As a matter of fact, it is notable that TeX's original definition for adjacent demerits is *not* completely orthogonal to the other parameters in the KP, and it is an exception. Indeed, the way adjacent demerits are applied does not depend only on the LSAR difference between consecutive lines, but on the LSARs themselves.

In Section 7, we have studied the impact of our refinement on the deviation from natural spacing. We have seen that as expected, the original algorithm pays more attention to it, although the average NRMS remains practically the same with our refinement in place. One thing we have yet to do is study the impact of our refinement on other aspects already handled by the KP. In particular, does our refinement induce more hyphenation, or worse, more hyphenation ladders? Does it consistently lead to longer or shorter paragraphs? What is the impact on similarities if we also use the extension proposed in [18]? These are important questions that, once answered, may lead us to adapt the default values of several parameters in the KP in order to restore the balance.

Finally, the hypotheses we have formulated as to what participates in our perception of homogeneousness are admittedly conjectural. They are based on the analysis of one particular example, intuition, and a pinch of introspection. We would very much like to confront these ideas with rigorous experimentation in cognitive science and psychology of perception. There is literature about how typography affects the process of reading [5, 10, 14], but most of it seems to be focused on the perception of letterforms, word recognition, and the design of typefaces (in particular, consistency and uniformity). We are not aware of any recent research about what makes the quality of the typographic color from a cognitive and perceptual point of view. Perhaps there is a whole new field of research to develop there.

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] 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.
[4] Alex Holkner. *Global Multiple Objective Line Breaking.* PhD thesis, School of Computer Science and Information Technology, RMIT University, Melbourne, Australia, October 2006.
[5] Daniel Janini, Chris Hamblin, Arturo Deza, and Talia Konkle. General object-based features account for letter perception better than specialized letter features. *Computational Biology*, 2021. doi: 10.1101/2021.04.21.440772.
[6] Eric A. Kenninga. Optimal line break determination. US Patent 6,510,441, January 2003.
[7] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting.* Addison-Wesley, MA, USA, 1986. ISBN 0201134470.
[8] Donald E. Knuth. *TeX: the Program*, volume B of *Computers and Typesetting.* Addison-Wesley, MA, USA, 1986. ISBN 0201134373.
[9] 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.
[10] Clive Lewis and Peter Walker. Typographic influences on reading. *British Journal of Psychology*, 80(2):241–257, 1989. doi: https://doi.org/10.1111/j.2044-8295.1989.tb02317.x.
[11] Frank Mittelbach. E-TeX: Guidelines for future TeX extensions – revisited. *TUGboat*, 34(1), 2013.
[12] 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.
[13] 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.
[14] Thomas Sanocki and Mary Dyson. Letter processing and font information during reading: Beyond distinctiveness, where vision meets design. *Attention, perception & psychophysics*, 74:132–45, 01 2012. doi: 10.3758/s13414-011-0220-9.
[15] H. T. Thành. Micro-typographic extensions to the TeX typesetting system. *TUGboat*, 21(4), 2000.
[16] 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.
[17] 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.
[18] Didier Verna. Similarity problems in paragraph justification: an extension to the Knuth-Plass algorithm. In *Proceedings of the ACM Symposium on Document Engineering 2024*, DocEng'24, pages 127–130, New York, NY, USA, August 2024. Association for Computing Machinery. ISBN 9798400711695. doi: 10.1145/3685650.3685666.