Abstract

In this paper we describe our experiments with paragraph optimization, based on some ideas from the *hz*-program. We extend the mechanism of paragraph composition of the TEX program in several ways, in order to examine the influence of various techniques on the grayness of the page and the readability of the composed text.

Introduction to TEX line breaking

Before we go into the details of our experiments, we will first describe briefly the mechanism that TEX uses to compose paragraphs (*line breaking* in TEX terms). The description is necessary because most of the issues in our experiments deal with paragraph composition.

The technical details of TEX paragraph composition are rather complicated, however the principal concept is quite straightforward. It is based on the *box/glue/penalty* model as described below.

- A *box* represents some material that should be typeset, usually a character or a sequence of characters from a font, but it can also be a much more complex object, e.g. a mathematical formula or a composition of boxes. For the purpose of paragraph composition we will treat boxes simply as words or segments of hyphenated words.

- A *glue* represents a blank space that has varying width in a specified way. For example the word space in text is usually a glue. A glue has its *natural size*, standing for the normal width of the glue. Apart from its natural size, a glue may have its *stretchability*, which is the maximum extra amount that the glue can be increased by. Similarly, glue *shrinkability* stands for the maximum amount that the glue can be decreased by. The natural size, stretchability and shrinkability together are called *glue specification*. In the context of paragraph composition, the word space is treated as a glue, with its specification depending on the used font.

- A *penalty* is the cost we pay or the reward we gain for breaking a line at a certain place. Using penalties thus allows the control of line breaking in a flexible way by specifying appropriate values of penalty at the desired places.

In addition to this model we have a *breakpoint* is a place where it is possible to end a line. A breakpoint usually occurs at a glue, a penalty or a *discretionary break*. A discretionary break is a place where a word can be hyphenated. During paragraph composition, TEX examines various breakpoints and tries to fit lines to the paragraph width.

When trying to fit a line to the desired width, the "natural width" of the line is examined first. This width is the total of box widths plus the natural size of the glues in the line. If the natural width is the same as the desired width, this line is treated as a "good" line and has the *badness* equal to zero. If the natural size of a line is shorter than the desired width, then the glues in the line will be stretched to adjust the width to the desired line. Similarly, a line longer than the desired width will be adjusted by shrinking the glues in the line. The more the glues in a line are stretched or shrunk, the higher *badness* the line will have. When a potential line is too short or too long, the difference between the natural width of the line and the desired width may be larger than the total stretchability or shrinkability of the glues in the line. In such cases the line is treated as "infinitely bad" and has *infinite* badness. Apart from badness, a line may have a penalty associated with the ending breakpoint of the line. The *cost* of breaking a line depends on the badness and the penalty associated with the line.

The main task of paragraph composition is to break a paragraph in such a way that the total *demerits* (which is roughly the sum of the squares of badness and penalty over all lines) is minimal. Therefore paragraphs composed by TeX have word spaces distributed approximatively equally over all lines of the paragraph rather than for a single line. This mechanism to break a paragraph as a whole gives a much better result than breaking in sequence from line to line at a time, because in the latter case, setting optimal word spacing in a line may cause very bad word spacing for the next lines.

It has been shown in practice that TeX line breaking works very well in most cases. However, we think that improvements are still possible, especially for difficult cases like in narrow column typesetting.

The hz-program

The *hz*-program, named after the inventor Hermann Zapf, tries to achieve even grayness of paragraph typesetting by keeping word spaces unchanged or changing them very slightly. In order to do this, the *hz*-program applies some techniques that have been known among typographers for years, namely skillful kerning, scaling and composing. The implementation is divided into several modules.

- Kerning is handled by the *kf*-program. Kerning is calculated on the fly depending on character shapes and pointsize, and can be done in the context of more characters than just adjacent characters (for example, kerning for all characters in a word or a group of words). Another unique feature of the *kf*-program is the ability to deal with side bearing of characters at the beginning or the end of a line in order to keep the margin of text more optically straight.

- Scaling is done within the *Kϱ*-program, which provides optical scaling of character letterforms.
- Font expansion and condensing can be done without changing stroke widths of letterforms by the *Ɛk*-program.
- Paragraph composition is done by the *jp*-program using the similar algorithm as inTEX.

There is not much documentation available about the *hz*-program. Perhaps the most complete source is the brochure about the *hz*-program by URW. This is also the source of our inspiration in doing the experiments. Our implementation prototype is pdfTEX, an extension of TEX. The experiments are done in several steps and we will describe in sequence what we are experimenting with in each step.

The basic sample set

In our testing, we used 3 types of fonts: METAFONT, Multiple Master and Type 1. From each type we chose two fonts, one roman and one sans serif. The following table shows the fonts used in our experiment:

	METAFONT	Multiple Master	Type 1
roman	Computer Modern Roman	Minion	Palatino
sans serif	Computer Modern Sans Serif	Myriad	Helvetica

Samples for each font are made at 3 sizes: 8pt, 10pt and 12pt. CMR and CMSS fonts have design sizes for all of these sizes, so the fonts at the design size were used. Multiple Master Minion has an optical size axis, and we used instances at the requested size to get the best optical scaling. For the Multiple Master Myriad and Type 1 fonts, one single font was scaled for all sizes. The used text in samples are from *"The tale of a youth who set out to learn what fear was"* by the brothers Grimm. Samples at different font sizes have different column setting, as shown in the following table:

	8pt	10pt	12pt
number of columns	5	4	3
column separator	3 mm	4 mm	5 mm
column width	33 mm	42 mm	56 mm

The bottom line of each sample reports to the set the sample belongs to, the font name and font size used in the sample, plus extra settings applied to the current set.

The first set contains the basic samples. The parameters affecting paragraph typesetting in this set are used for the next steps as well. The following settings have been tuned to get rid of so-called overfull boxes, which are lines that are longer than the desired line length of the paragraph.

- Paragraph indentation is set to 1em;
- Leading is set to 1.2em;
- Column heights are adjusted for each sample to avoid orphans and widows as well as to balance columns;
- No extra space is added between paragraphs;
- No extra space is added after punctuations;
- In extremely difficult cases, a line is allowed to be longer than the desired width by as much as 1pt;
- In extremely difficult cases, the total amount of all word spaces in a line can be increased by as much as 2pt;

In TeX terms this is expressed by:

```
\parindent=1em                    \parskip=0pt
\global\baselineskip=1.2em        \emergencystretch=2pt
\clubpenalty=10000                \hfuzz=1pt
\widowpenalty=10000               \tolerance=5000
\frenchspacing
```

The last setting `\tolerance=5000` is too complicated to describe in non-TeX terms. In short, it says that the word space can be changed (stretched or shrunk) by a relatively high percentage of its stretchability or shrinkability. This setting is necessary for narrow width typesetting, e.g. in the case of having multi-columns in a page.

Marginal kerning

In the second set we apply marginal kerning. This is indicated at the bottom line of the samples by MarginKerning=yes.

In the *hz*-program, an artificial character “left white edge” is placed to the left of the beginning character in a line, so the program can calculate the left side bearing for the leftmost character. Similarly a “right white edge” character is used for the rightmost character. Those two characters don’t exist in reality, but they are only generated by the program on the fly in order to control marginal kerning.

In pdfTeX, marginal kerning is achieved by using a mechanism called *character protruding*. Each character has an associated parameter called *left protruding factor*. This parameter specifies how much the character should "protrude out" to the left if the character ends up at the beginning of a line. The amount of protruding is given in thousandths of the corresponding character width. For example, a left protruding factor of 500 of a character says that the character should protrude out to the left by 50% of its width. Similarly, the *right protruding factor* is used for protruding characters at the end of a line. These parameters can be negative as well, which means "protruding in" the opposite direction. In pdfTeX terms, \lpcode is used for left protruding factor and \rpcode for right protruding factor.

The setting of protruding factors in our experiments are listed below (in TeX terms the notation ‘\ indicates the numeric code of the following character, i.e. ‘\A is equivalent to the ASCII code of the character A, which is 65).

```
\rpcode`\!=200        \rpcode`\F=50         \rpcode`\y=50
\rpcode`\,=700        \rpcode`\K=50         \lpcode`\(=50
\rpcode`\-=700        \rpcode`\L=50         \lpcode`\A=50
\rpcode`\.=700        \rpcode`\T=50         \lpcode`\J=50
\rpcode`\;=500        \rpcode`\V=50         \lpcode`\T=50
\rpcode`\:=500        \rpcode`\W=50         \lpcode`\V=50
\lpcode`\`=700        \rpcode`\X=50         \lpcode`\W=50
\rpcode`\'=700        \rpcode`\Y=50         \lpcode`\X=50
\lpcode92=500  % ``   \rpcode`\k=50         \lpcode`\Y=50
\rpcode34=500  % ''   \rpcode`\r=50         \lpcode`\v=50
\rpcode123=300 % --   \rpcode`\t=50         \lpcode`\w=50
\rpcode124=200 % ---  \rpcode`\v=50         \lpcode`\x=50
\rpcode`\)=50         \rpcode`\w=50         \lpcode`\y=50
\rpcode`\A=50         \rpcode`\x=50
```

As we can see from this list, the most prominent settings are for punctuations. However there are also slight protrudings of certain characters which have a lot of white area on left-hand or right-hand side of their shapes.

The values of protruding factors are used in two phases. In the first phase when pdfTeX tries to fit individual lines to the desired width, it also checks whether the beginning or the final object in a line is a character with non-zero protruding factor. If so, their protruding amount is added to the desired width before pdfTeX calculates the badness of the line. In the second phase, after the breakpoints for the paragraph have been found, pdfTeX inserts the corresponding kerns to the required places before dividing the paragraph to lines at found breakpoints.

Applying marginal kerning causes that the desired width for a line may be adjusted by the protruding amount of the beginning and/or the ending character of the line.

Therefore the badness calculation and the chosen breakpoints may be changed. For this reason, it is possible that a paragraph will be broken in a different way than TEX usually does, with different total demerits. The variance of total demerits can be positive as well as negative, however it is is not very large in percentage. Roughly speaking, applying character protruding has similar effect to total demerits calculation as in case the average desired width is changed a little bit.

Font expansion and condensing

In the third set we apply font expansion and condensing[1] to TEX paragraph composition. To enable it, a font must be first specified to be "expandable". The limits of stretching and shrinking can be given separately, so a font can be stretched only (by giving a non-zero limit of stretching and zero limit of shrinking), shrunk only or stretched and shrunk with different limits. A font is not expanded to an arbitrary amount, but rather by certain discrete steps up to the limit of expansion. The limit and the step of expansion are given in thousandths of the font width. For example, a font with stretchability 50, shrinkability 40 and expansion step 5 means that the font can be stretched by 0.5%, 1%, 1.5%,. . . , 5%, and can be shrunk by -0.5%, -1%, -1.5%,. . . , -4% of its width. What "a font stretched by 1% of its width" exactly means is depending on the type of the font. Roughly speaking, it could be understood as a font with the average width of characters increased by 1%.

At the phase of looking for breakpoints, pdfTEX "expands" a font by loading the font with the same name as the name of the expandable font but with a number appended. This number specifies the expansion amount (in thousandths of the font width as described above). For example, when a font, named `cmr10` (Computer Modern Roman at design size 10pt), is expanded by 1%, pdfTEX will try to load a font with name `cmr10+10`. Similarly `cmr10-20` is loaded for -2% expansion. The higher the limit and the smaller step is used, the more fonts will be loaded. When pdfTEX cannot find such an expanded font, the font will be created on the fly. The way an expanded font is created depends on the "real" type of individual fonts and is done outside pdfTEX. Font expansion can be applied to various types of fonts and will be discussed below.

When an expandable font is searched, pdfTEX loads into memory the metrics expanded at the limit of expansion. This is needed to calculate the maximum amount that can be adjusted for character widths from the expandable font. For example, when a font named `cmr10` with stretchability 50 and shrinkability 40 is searched, pdfTEX will load fonts `cmr10+50` and `cmr10-40`. Then the width of a character from font `cmr10` can be mostly "stretched" to the width of this character from the font

[1]From now on, we will use expansion for both expansion and condensing. Condensing can be treated as expanding by a negative amount. We will use stretching and shrinking when it is necessary to distinguish the effects of expansion.

`cmr10+50` and “shrunk” to the character width from font `cmr10-40`. This is an analogy to glue stretchability and shrinkability, so we call it *character stretchability* and *character shrinkability*.

Before pdfTeX stretches a line that is shorter than the desired width, it also checks whether the total character stretchability of the line is non-zero. If so, the total character stretchability is used to decrease the difference between the natural line width and the desired width. Similarly, the total character shrinkability is used in the case of shrinking.

Similar to protruding factors, character stretch and shrink are used in two phases: when trying to fit individual lines to the desired width and when dividing a paragraph into lines at found breakpoints. After the breakpoints in a paragraph have been found, expandable fonts in individual lines are substituted by expanded fonts, which have character widths *different* from the original expandable fonts. The difference between total character widths of expandable fonts and corresponding expanded fonts in a line is exactly equal to the total character stretchability or shrinkability that has been taken into account for the line while pdfTeX was trying to fit the line into the desired width.

The most important factor for font expansion is to keep the stroke widths of letterforms unchanged. Therefore the characteristics of each font type that may have an effect on font expansion are important to us.

- Computer Modern fonts were developed by Donald Knuth using a special tool called METAFONT, which is a companion program to TeX. Apart from having many nice features, Computer Modern fonts are interesting for our experiments in the sense that they are generated from the same sources with various settings of parameters independent of each other. In particular, Computer Modern fonts can be expanded without changing stroke widths of letterforms. This is done by altering a parameter called *width unit* in METAFONT terms. A Computer Modern font is expanded by increasing the width unit by the requested amount, i.e. `cmr10+10` is created by increasing the width unit by 1% of the original unit width. The characters of Computer Modern fonts are expanded linearly, which means that they are wider by exactly the required amount. E.g. a font expanded by 2% will have every character width wider by 2% than the original character width.

- Multiple Master fonts were developed by Adobe. A Multiple Master font can have two or more design axes. A design axis represents a dynamic range of one typographic parameter, which is usually weight, width or optical size. A Multiple Master font can be used to generate various Type 1 fonts, called *instances*. To create an instance, valid values must be given to all design axes. For our experiments, Multiple Master fonts with width axis are interesting, as they can be expanded without distortion of the letterforms. We expand a Multiple Master instance by changing the width value used to generate the instance. For example, the Multiple Master instance of the font Minion used in our samples has the width value of 535, thus expanding a font by 2% is done by creating a new instance with the width value $535 + (20/1000)\times 535 = 545.7$. It is important to know that characters of Multiple Master fonts are *not* expanded linearly, but they change differently for different characters. The change depends rather on individual character shapes. Normally, uppercase letters and non-letter characters are expanded much more than lowercase letters. In comparison with Computer Modern fonts, usually a Multiple Master instance that is expanded by the same amount as a Computer Modern font has smaller changes in character widths (in percentage of character width), but in some rare cases these changes can be larger as well (mostly cases of uppercase letters).

- Type 1 fonts, developed by Adobe, were not designed to be expanded. However, it is quite common that these fonts are expanded by applying horizontal scaling. This technique causes every part of the letterforms to be scaled, so the stroke widths are changed as well. Given that fact, theoretically Type 1 fonts are not suitable for our purpose at all. However, at a very small amount of expansion, the letterforms distortion cannot be seen at all. Therefore it makes sense to apply the optimization using Type 1 fonts as well. Type 1 font expansion has similar characteristics to Computer Modern font concerning changes of character widths, i.e. they are changed linearly.

We do not apply any internal skillful kerning and letter spacing to the fonts as the *kf*-program does. Perhaps this is the main shortcoming of our experiments. We rely on the fact that kerning and letter spacing are changed correspondingly when a font is expanded. Therefore, by using the new metrics we expect the desired effect on kerning and letter spacing as well. This is of course far from the optimal solution. However, we don't have any information available to implement the capabilities of the *kf*-program into our prototype.

In the samples we use the term FontExpand=⟨number⟩ (shown at the bottom line of each sample) to indicate that font stretch and shrink is applied to the current set. Both font stretch and shrink are set to the value of FontExpand. The third set of samples was generated with such limits on font expansion so that the distortion of letterforms is not visible yet. In our experiments, the reasonable limit of font expansion is about 20 for Computer Modern and Type 1 fonts. For Multiple Master fonts, the limit is about 40. It seems that Type 1 and Computer Modern fonts can be expanded by nearly the same limit, though Computer Modern fonts can be expanded without changing the stroke widths of letterforms. We think that the limit of font expansion strongly depends on characteristics of individual typefaces. Choosing a reasonable value for a typeface requires a lot of experimenting with the value of limit of expansion as well as further parameters that influence word spacing in a paragraph.

In the fourth set we increase the limit of font expansion a little bit more. In some cases the difference between fonts used in individual lines is becoming visible. We consider the setting in this set as the limit that should not be exceeded anymore.

Expansion of selected characters

In the next step we experiment with another idea of the *hz*-program that font expansion should be applied to selected characters rather than to all characters. For this purpose we introduce a new parameter called *expansion factor* (`\efcode` in pdfTeX terms) associated with each character. This parameter says how many thousandths of stretchability and shrinkability of a character can be used. The valid range of expansion factor is from 0 to 1000. By default all characters have expansion factor of 1000. E.g. a character with expansion factor 300 will be expanded as much as 30% of other characters with expansion factor of 1000. Therefore, if in a line all characters with expansion factor of 1000 are expanded by 1.5% of their width, then the character with expansion factor of 300 will be expanded by 0.5% of its width. The purpose of this mechanism is to limit expansion factor of characters that are more sensitive to expansion.

By experiments we found the following cases to be more sensitive to font expansion (even without changing the stroke widths of letterforms).

1. Characters with long and strong strokes in horizontal direction (not only those parallel with the baseline). When the character width is changed, those strokes often make the character shape darker. Typical examples for such characters are A, K and R.

2. Characters with white areas in letterform bounded by strokes in vertical direction (need not to be upright to the baseline). Those areas also have a strong influence on darkness of letterform and therefore make the character sensitive to expansion. There are many of such characters, like o, u, e, p and q.

3. Letters are more sensitive to expansion than digits and non-letter characters.

Based on these factors, we adjusted the expansion factor in this step as the following:

```
\efcode`\2=700        \efcode`\B=700        \efcode`\H=700
\efcode`\3=700        \efcode`\C=700        \efcode`\K=700
\efcode`\6=700        \efcode`\D=500        \efcode`\M=700
\efcode`\8=700        \efcode`\E=700        \efcode`\N=700
\efcode`\9=700        \efcode`\F=700        \efcode`\O=500
\efcode`\A=500        \efcode`\G=500        \efcode`\P=700
```

```
\efcode`\Q=500          \efcode`\c=700          \efcode`\n=700
\efcode`\R=700          \efcode`\d=700          \efcode`\o=700
\efcode`\S=700          \efcode`\e=700          \efcode`\p=700
\efcode`\U=700          \efcode`\g=700          \efcode`\q=700
\efcode`\Z=700          \efcode`\h=700          \efcode`\s=700
\efcode`\a=700          \efcode`\k=700          \efcode`\u=700
\efcode`\b=700          \efcode`\m=700          \efcode`\z=700
```

Using adjusted expansion factors is indicated by SelectGlyphs=yes at the bottom line of the samples in the fifth set. The values for the limit of expansion are taken from the fourth set (with settings of font expansion that is becoming visible in some cases).

Letter spacing

After applying the previous techniques, we found out that even when differences between letterforms are not recognizable at all, one can still recognize that some lines are much darker or lighter than others. This has to do with the fact that given a line with the desired width, our goal is to change word spaces as little as possible. This happens when all interword spaces are set equal or very close to their natural size. In order to achieve this, the extra amount of space (the difference between natural line width and the desired width) must be divided into some other kind of space consuming entity, like expansion of certain characters or changing letter spacing. Therefore, in any case the darkness of the line seems to be unchanged, because the proportion of the total amount of "black strokes" and "white areas" seems to be constant.

Taking the above into consideration in the next steps, we tried not to expand characters at all, but change the space between adjacent characters instead. This technique, known as adjustment of character spacing or tracking, is widely used even though it is not encouraged by many typographers. However, for our purpose, we think that distributing the extra amount of space between individual characters (i.e. changing character spacing) is acceptable when applied in very small quantities and as part of a sophisticated line break algorithm as present in TeX.

To examine this hypothesis we repeated all experiments with font expansion in set 3–5, using the same limits and expansion factors. This time, however, when the calculation for putting the characters on the page has been done, we use non-expanded characters instead of expanded characters to draw the output. Therefore, each character from expanded fonts will be replaced by the corresponding non-expanded character, followed by a little space (can be either positive or negative) at the right bearing side. The more a character from a font is expanded during line breaking, the larger the amount of the space at the right bearing of the character will be. This technique is indicated by ExpandGlyphs=no at the bottom line in sets 6–8, It is possible even to replace expanded characters by corresponding characters from another font with smaller expansion, i.e. it is possible to “scale down” the expansion amount of individual characters after all calculation for line breaking has been done. Each expandable font has an associated parameter called *scale factor*, which says how much expansion of characters from that font should be scaled down. Scale factor is given in thousandths. In sets 3–5 all fonts have been used with scale factor 1000, which stands for “full” expansion. On the contrary, the fonts in set 6–8 have been used with scale factor 0, causing that all expanded glyphs are scaled down to non-expanded ones (zero expansion). A value of 500 means that if a character has been treated as expanded by i.e. 2% while breaking lines, in the final output it will be expanded by only 1%.

During line breaking, the scale factor is not used at all, because pdfTEX does all calculation with the expanded metrics regardless of the scale factor. Therefore the scale factor of a font does not influence line breaking, it has effect only on how individual characters will be drawn after the breakpoints have been found. Sets 6–8 have exactly the same result of line breaking as sets 3–5, including the total demerits of individual paragraphs.

In sets 6–8 there is no difference between letterforms in individual lines in a page. However the distortion of text seems to be increased in comparison with sets 3–5. We consider changing letter space to have more negative influence on readability and beauty than changing letterforms. For this reason we decided not to make any further samples with other values of the scale factor.

Demerits comparison

As described above, the goal of TEX line breaking is to divide a paragraph into lines in such a way that the total demerits is as small as possible. The smaller the total demerits a paragraph has, the “better” TEX treats the paragraph. In other words, the criteria how TEX evaluates line breaking is the value of total demerits. In order to examine whether the applied techniques does help to improve TEX line breaking at all, we keep track how the total demerits changes for individual paragraphs in each step.

For each sample, we created a table showing how the total demerits of the first 8 paragraphs changed in sequence in sets 1–5. Sets 6–8 have the same total demerits as set 3–5, so they don’t need to be listed here.

CM Roman at 8pt

	1	2	3	4	5	6	7	8
1	32961945	11866820	21463929	15231528	5428036	62106976	22734505	32798450
2	20649021	17293395	24404188	15202590	5810757	85140299	18222047	36845028
3	9926418	7518113	7348730	3635661	3699529	17591719	66673527	19573369
4	1664022	5734599	2860032	2037011	1701044	3934010	11632314	10441436
5	8652650	6812474	5944942	2833829	3159516	13265822	15191508	14847043

CM Roman at 10pt

	1	2	3	4	5	6	7	8
1	51480729	16351262	61219352	31014102	21555042	14845292	31288136	93734014
2	68072688	7688624	51840565	33994661	22748997	18783697	77452773	111284279
3	9451448	2979202	10719346	1611282	11648977	4968898	20816378	22832668
4	6198947	1811992	2720130	1028823	8421115	24643916	14092901	11301111
5	8161201	2429543	3609299	1391795	10280968	4465918	17904696	17749166

CM Roman at 12pt

	1	2	3	4	5	6	7	8
1	27455985	23739375	12809490	3367856	6244656	9202903	13874638	10761008
2	19004489	2669852	10705954	1364531	6655303	8223644	19033739	12776017
3	7662280	174065	933384	503868	1811755	823283	1162904	841764
4	207171	76091	395286	298405	1049397	243728	164617	494488
5	4326643	137892	806170	402116	1441380	616550	670881	665368

CM Sans Serif at 8pt

	1	2	3	4	5	6	7	8
1	17060265	5970172	63025143	6589353	50104582	66858091	16728655	59699767
2	18830826	6146405	51056033	6638093	52742491	48355024	18092694	53933110
3	4498520	1677844	10564024	2789277	2397052	14964251	13490005	10511947
4	2759324	227821	1129922	1867162	1557226	6861928	3925343	4935540
5	3740822	555436	5403571	2406392	1997855	12457179	11081478	8802036

CM Sans Serif at 10pt

	1	2	3	4	5	6	7	8
1	23373572	11016109	20209877	12965133	24137434	10601297	27529612	40977090
2	31891286	2832377	19190485	529514	24528167	15452207	30896432	50329234
3	4155743	1095327	3824077	182576	752429	753006	14874377	2167549
4	2595845	224278	460530	108259	449649	269542	3447507	565054
5	3535249	391513	1157929	148928	607515	466387	11942516	1406684

CM Sans Serif at 12pt

	1	2	3	4	5	6	7	8
1	4915097	1201586	5902683	1773797	10610761	3509219	4112100	5881029
2	5180779	1755625	5981881	1775855	10229894	3984009	5826672	5850458
3	1760025	38092	711236	70273	5567457	219251	960832	917710
4	758632	18937	308280	26899	23116327	57066	473102	182523
5	1502574	34760	494633	61586	4891943	113836	702369	355602

Minion at 8pt

	1	2	3	4	5	6	7	8
1	1088621	2431320	2645634	221430	558984	1343846	4649977	2326310
2	1517734	225192	823380	281484	635258	1459725	4651600	2819580
3	485760	95385	203328	151208	126255	372753	2165887	457026
4	391658	76449	157634	131316	107215	237475	1765358	322037
5	512875	97476	207987	152891	127097	394561	2207325	476934

Minion at 10pt

	1	2	3	4	5	6	7	8
1	3433166	498433	850647	866825	1187328	1214217	1463205	2727425
2	1991498	438201	651771	1142484	1335398	1295794	1726983	2353404
3	116952	80550	170196	590925	433392	213682	599521	269901
4	97964	68415	131832	496409	298456	136589	193567	212727
5	120650	80783	263636	584962	423806	216135	345459	272085

Minion at 12pt

	1	2	3	4	5	6	7	8
1	711190	321601	493524	66119	149909	490401	484901	751395
2	775702	336090	415659	68228	197120	558958	805514	880535
3	71658	41299	57896	25296	25892	51674	128026	92174
4	58519	32379	45818	19929	22886	41572	89287	71538
5	73173	40598	58832	26087	26427	53034	130298	102843

Myriad at 8pt

	1	2	3	4	5	6	7	8
1	3038531	410191	795345	479000	2096379	1340027	2813763	1988757
2	2290485	382634	839314	561483	1858459	1251748	2216402	1253868
3	81879	77401	254967	250959	826639	206869	496832	222438
4	68420	45169	166787	212180	660122	152409	340572	126957
5	84158	78935	270053	243268	828658	208389	482906	241645

Myriad at 10pt

	1	2	3	4	5	6	7	8
1	439940	141412	1383610	1108889	3962678	1083938	3804137	1056041
2	426043	178846	1411259	1323899	3914016	959934	2545765	1097451
3	211626	92388	232689	22615	2039468	134571	438498	295911
4	150559	80116	174603	16365	1694971	71048	294824	210927
5	214961	91531	268783	24388	2055045	139162	437766	301848

Myriad at 12pt

	1	2	3	4	5	6	7	8
1	984509	210315	397315	68078	193596	213415	1390396	916878
2	1031217	159440	475628	58856	198906	189304	961489	499023
3	74175	50807	69269	49898	29564	61315	91195	76214
4	45125	40368	55495	21790	10782	42455	65295	54993
5	86806	50961	70384	49211	30392	65154	94903	78116

Palatino at 8pt

	1	2	3	4	5	6	7	8
1	10962748	10218286	30554260	9789030	22824637	29270656	14547596	36597253
2	10603045	2380937	22882159	4665403	27116384	33440378	49779147	43125078
3	3356858	1042246	6011245	1409620	1903209	6269031	18768587	8674929
4	417625	676173	1882699	735082	1350905	4392122	4578060	3337301
5	739060	914457	2759121	1151627	1654422	5581451	16089063	5964059

Palatino at 10pt

	1	2	3	4	5	6	7	8
1	14289306	5039876	27036222	11371553	6919342	28211864	66801793	52985513
2	18409958	5447105	26975473	11685206	7958355	25623877	76668341	44762495
3	1912662	967891	6235247	4944495	693621	13333630	13102104	7919949
4	1220919	599671	1205939	2059779	414773	8172922	6973894	4765055
5	1686932	786073	1548567	4141428	600512	9951608	10422254	6790359

Palatino at 12pt

	1	2	3	4	5	6	7	8
1	29631797	3984097	8623401	215090	669750	4338275	16502949	16678749
2	17665073	6074973	7317302	273119	711200	5277127	17207975	17993702
3	8670689	146715	802045	100184	270185	409386	2563446	1497360
4	661673	91314	149294	71760	107617	212487	1388392	483533
5	7396945	124885	678447	86893	209682	350820	1793228	1326187

Helvetica at 8pt

	1	2	3	4	5	6	7	8
1	23349628	3835003	26633465	15724044	7446470	19673533	12324121	42290333
2	17932830	3993801	35692490	7607037	8242835	13570147	14387186	51022501
3	2630397	1257535	3849759	108370	376470	6128726	15401971	5832377
4	709864	748351	1706956	58509	248508	4408026	6237467	2089158
5	2286100	1060654	2497222	90260	319547	5489705	8046918	3498542

Helvetica at 10pt

	1	2	3	4	5	6	7	8
1	14499298	6664853	21491275	303298	2493481	23613219	49109211	31012167
2	14246523	7246356	26108371	746848	2311044	23850290	63531794	29174492
3	1057444	3356738	2469114	287715	678468	13207014	7012584	7076586
4	673359	973108	1164525	172720	413148	1329194	4062876	2318725
5	916798	2908297	2075792	242087	577702	11939761	6198727	5819144

Helvetica at 12pt

	1	2	3	4	5	6	7	8
1	18176601	4943074	5825636	897222	2330404	3140966	14186963	9505728
2	18574661	1110776	2890829	1016422	2350857	2014190	15417016	6919577
3	8644904	168558	507415	372125	737252	470536	2369455	400669
4	5870989	124498	155594	171744	420760	208942	407985	158978
5	7619280	151361	393085	218404	594224	372252	2019615	268057

As it can be seen from the tables, the total demerits for each paragraph changed in all steps in a very similar way.

1. As one could expect, applying marginal kerning doesn't change the total demerits very much, even though the breakpoints can be found in a different way.

2. Set 3 has much smaller total demerits, as the word spaces need not to be stretched or shrunk as much as in set 1 and 2.

3. Set 4 has smaller total demerits in comparison with set 3, since the more a font can be expanded, the less the word spaces have to be stretched or shrunk.

4. Set 5 has larger total demerits than set 4 even though the value of font expansion is the same, because in set 5 some characters can be expanded by smaller amounts than in set 4. The total character stretchability and shrinkability is smaller, which leads to larger badness of lines and therefore larger total demerits.

Remarks

To draw conclusions is not easy, since people tend to value the quality of a typeset document differently. While top quality typographers seem to agree on aspects like optimal grayness and what kind of manipulations are permitted and what not, the average desk top publisher is very tolerant in applying generous spacing and seemingly arbitrary stretching of glyphs and kerns to achieve his personal objectives. Since the focus of this work is on the computer science side of the problem, we leave the more subjective conclusions to the ergonomist. On the other hand, we think it is important to summarize our remarks during the experiments and ask the experts to judge of our opinions. We consider it to be necessary to get feedback on our experimentation as well as suggestions for further development.

1. Marginal kerning does not influence interword spacing and does not damage the legibility of composed text at all. It seems to be the most visible improvement and moreover, it can be used without any extra setup. Therefore it is safe (and good) to apply this technique to paragraph composition.

2. Font expansion does help to make word spaces more even. On the other hand it also impairs text readability. Limits of font expansion depend on many factors and must be chosen very carefully for individual fonts. Very roughly speaking, the limit of font expansion shouldn't be more than 50 for Multiple Master fonts and 30 for METAFONT as well as Type 1 fonts. Expansion of selected glyphs may help to decrease the visual effect of font expansion (the difference between glyphs from a font expanded at various values).

3. Even word spaces is not the only factor to uniform grayness of composed text, since the darkness of a line depends on the letterforms of characters in the line as well as letter space. Moreover, letter space seems to be more sensitive to darkness and legibility than letterforms. Therefore we think that in order to get better results, a program which can do skillful kerning on the fly like the *kf*-program is a must.

4. Even word spaces do not guarantee that there will be no rivers in composed paragraphs. Rivers can also appear in cases when word spaces are set to be very even, because rivers are formed rather by position than size or uniformity of word spaces. Therefore it would be very useful to integrate a "river detector" into the mechanism of line breaking.

5. Darkness of a line is depending not only on word spaces, but also on darkness of individual letterforms in the line. Moreover, equal word spaces on a line need not to look optically uniform because of the white area in side bearing of characters adjacent to the word spaces. For this reason, some fonts might contain kerning pairs between the character space and other characters. Although these kerns could be used with TEX, in practice they are often ignored.[2]

6. In order to achieve more uniform grayness of text, we think that it might be worth to introduce another model of badness calculation for line breaking. Instead of using the measure of stretching or shrinking word spaces in a line, the darkness of the line could be used as the main factor for line breaking. The darkness of a line can be calculated according to

   (a) the number of word spaces in the line,

   (b) the measure of stretching or shrinking word spaces in the line,

   (c) the darkness of individual characters in the line,

   (d) the context of characters and word spaces in the line.

   Darkness of a character could be calculated on the fly, or better an external program can be called to generate darkness for all characters from a font ahead. This of course could be influenced by font expansion if the font is "expandable". The darkness of a line can be also altered by skillful kerning of a group of words, which should take the above thoughts into consideration as well. Like the *kf*-program, adjustment should be done not only for word spaces but also for letter spaces.

   Then given a particular line, the cost of the line for paragraph breaking is depending on the difference between the darkness of the line and the "average darkness". The goal of paragraph composition would be to minimize the total differences of darkness over all individual lines of a paragraph. In other words, the line breaking algorithm would try to make all lines of a paragraph have more uniform darkness instead of interword spacing.

[2]It has to do with the concept of TEX that it does not use the character space at all. Spaces from input are replaced by glue with specifications read from the current font. However it is possible to kern a character adjacent to a space using the so-called *boundary character*.

We think that the mechanism described above would be realizable in pdfTEX if a module equivalent to the *kf*-program is available. At the moment, however, implementing such a module is out of scope of our work.

Conclusions

We are not experts in typography and therefore we do not dare to claim that the techniques we applied in our experiments improve things at all. The result of smaller total demerits of line breaking is not enough to prove that the output looks better. Our main goal in this paper, however, is to describe what we are experimenting with and what is possible to do with our prototype. We would like to get comments on our experimentation, as well as to invite people who are interested in this topic to get involved. We consider help from typographic experts to be the key for further development of our work.