ELECTRONIC PUBLISHING, VOL. 8(2 & 3), 139–152 (JUNE & SEPTEMBER 1995)

Pagination reconsidered

ANNE BRÜGGEMANN-KLEIN
Technische Universitt München
Fachbereich Informatik
Arcisstr. 21
80290 München, Germany

ROLF KLEIN AND STEFAN WOHLFEIL
FernUniversität Hagen
Praktische Informatik VI
Elberfelder Straße 95
58084 Hagen, Germany

e-mail: brueggem@informatik.tu-muenchen.de, rolf.klein@fernuni-hagen.de, stefan.wohlfeil@fernuni-hagen.de

SUMMARY
We present a new algorithm for pagination that minimizes the number of page turns that are necessary while reading a formatted document. This approach keeps the total number of pages small and places figures close to their citations. Examples show that the resulting documents are superior in quality to what standard formatting systems achieve. Our algorithm is easy to implement and runs in time proportional to the number of text objects times the number of floating objects.

KEY WORDS Document processing Formatting Pagination Page make-up Page breaking Floating objects

1 INTRODUCTION

Whereas today's formatting systems provide acceptable or even excellent line breaking quality, the quality of their page breaking leaves much to be desired. The problem is in positioning the floating objects such as figures, tables, and footnotes. Each of them should appear closely after its citation; that is, the text object from which it is first referred to. At the same time, underfull pages should be avoided. Since objects are separated by white space whose size depends on their types, a text object followed by another text object may not fill a page, whereas it does if followed by a floating object of the same height. This further complicates the pagination task.

In word processing systems such as *Microsoft Word*, floating objects are not even known. *Framemaker* [1] knows them but does not guarantee that they appear in their true order of citation. In Frame 5 for example a large figure cited within a line in the middle of a page p is placed on the next page if it doesn't fit. If two lines later a smaller figure is cited that fits on page p it is placed there. LaTeX maintains the correct order, but for positioning the floating objects a simple first-fit strategy is employed. It places a floating object, once it has been cited, on the first page where it fits.[1]

Professional designers and book production specialists are not satisfied with the quality of documents formatted by first-fit algorithms. They feel—and rightly so—that humans, laying out the pages manually, can achieve better-quality pagination than anything an automated system provides them with so far.

[1] From now on we call all floating objects *figures*, for short.

CCC 0894–3982/95/020139–14

Received 1 April 1996
Revised 20 June 1996

This situation calls for optimization algorithms. But expectations are dampened by the theoretical results by Plass [2]; he proved that optimal page breaking is in general NP-hard, hence computationally intractable. It is conceivable that these results have so far discouraged further effort towards better pagination algorithms.

However, a closer inspection of Plass's work shows that one should not give up so soon. The point is that the complexity of optimal page breaking depends on how optimality is defined! The NP-hardness result holds for an objective function that adds up *the squares* of the numbers of pages each figure is away from its citation. Is this is a realistic optimization criterion? According to it, one would happily remove six figures from the pages where they are cited, in order to decrease the page difference of another figure from 4 to 3. We feel that such a change would rather deteriorate the document.

On the other hand, if one uses the sum of the page distances themselves, instead of their squares, the optimization problem becomes computationally tractable, as Plass has also proved.

In principle, nothing is wrong with using the linear distance in pages between a citation of a figure and the figure itself, as a measure of badness for the placement of a single figure. But in order to obtain a suitable overall measure, two points must be taken into account.

1. If minimizing the page differences is the only objective, documents are likely to have many and loosely filled pages. In fact, if we only want to minimize the page differences, it will often be preferable not to put another text block onto the current page, if this text refers to a figure which does not also fit onto the current page. This tends to produce lengthy documents.
2. When formatting a double-sided document, it is acceptable that a reference appears on one page and the figure on the adjacent page. Such page differences should not be counted.[2]

Both phenomena are taken care of if we use, as our objective function that is to be minimized, the *total number of page turns* that are necessary while reading the formatted document. This is either the total number of pages (minus 1) plus the sum of all page differences caused by citations that do not appear on the same page as their figure (for single-sided documents) or p DIV 2 plus the sum of all page differences caused by citations that do not appear on the page spread as their figure (for double-sided documents).

Sometimes the two goals—to minimize the total number of pages and the sum of the page differences—are not equally important. For example, many conferences threaten to reject submissions that exceed in length a certain number of pages. Or, for an instruction booklet it can be crucial that the figures and the text explaining them appear on the same page or on adjacent pages. Therefore we suggest to minimize the weighted objective function

$$\alpha \cdot (\text{sum of non-adjacent page differences}) + \beta \cdot (\text{number of pages} - 1),$$

where $\alpha,\beta \geq 0$. The weights α and β can be supplied by the user (perhaps within some range specified by the designer). The default setting is $\alpha = \beta = 1$.

The remainder of this paper is organized as follows. In Section 2 and Section 3 we state in more detail the input and the page model we are working with, and the solution we suggest. Recently, our optimization algorithm has been implemented. In Section 4 we

[2] Here the style guides [3][4] make an exception from the rule that a figure must not appear on a page previous to its citation.

describe the top level of the algorithm. The first practical results are very encouraging. In Section 5 we compare the pagination of (part of) an official document published by the government of Bavaria [5] produced by LaTeX with the layout our algorithm computes.

2 THE INPUT AND THE PAGE MODEL

The make-up or pagination step in book production assembles into pages several separately prepared types of material, such as the main text in galley form, the figures, the tables, and the footnotes. Pagination is notoriously difficult, since a number of competing rules have to be satisfied simultaneously. The classical rules are:

1. Each page must be perfectly full; facing pages must be balanced.
2. Each figure and each table must be on the same page (or page spread) as its citation.
3. Each footnote must start on the same page as its citation and must fall on consecutive pages.

The pagination process gains some flexibility from white space (for example from space between paragraphs or around headings and displays) that can within limits stretch or shrink. Nevertheless, the three rules just mentioned are often impossible to satisfy perfectly. Therefore, pagination procedures strive to satisfy them as well as possible.

We start our investigation with a simple framework, considering just two input streams for the pagination procedure, namely a text stream and a figure stream. The text stream consists of alternating lines and spaces

$$t_1, \tau_1, t_2, \tau_2, \ldots, \tau_{m-1}, t_m$$

and the figures stream consists of alternating figures and spaces

$$f_1, \phi_1, f_2, \phi_2, \ldots, \phi_{n-1}, f_n$$

In particular, the lines (and the figures) are ordered according to their occurrence in the text stream (the figure stream). Therefore, we say $t_i \leq t_j$ ($f_i \leq f_j$) if t_i (f_i) occurs earlier in the stream than t_j (f_j); that is, if $i \leq j$.

For pagination, only the vertical extensions of lines, figures, and spaces are relevant. We assume therefore, that each line and each figure x has a fixed height $ht(x)$ and that each white space x has a minimal height $htMin(x)$ and a maximal height $htMax(x)$. This is similar to TeX's glue model [6]. Spaces also have an attribute *legalPageEnd* that states, if a page may end in this space. Pages must end in spaces with this attribute set. This is used to prevent widows and orphans and to keep headings on the same page with the beginning of the following paragraph.

Finally, we need to know in which piece of text each figure is cited. Therefore, a reference function

$$R: \{f_1, \ldots, f_n\} \longrightarrow \{t_1, \ldots, t_m\}$$

is also part of the input. The reference function specifies, for each figure, the line which contains the first citation of that figure. We admit only those reference functions for which the figures appear in the same order as their citations; that is, if $f \leq f'$ then $R(f) \leq R(f')$, for all figures f and f' in the figure stream.

Our page model is also very simple. First of all, we assume that all pages have the maximal height *htMax* and the minimal height *htMin.* These two parameters are part of the design specification for the document; they form input parameters for the pagination routine. A design specification that sets *htMin* as strictly smaller than *htMax* allows that pages are not completely full, but calls for a minimal degree of fullness of

$$\frac{htMin}{htMax} \cdot 100\%.$$

Finally, we assume that the group of figures and the group of lines of text that are placed together onto a single page are divided by some white space, whose extension is also part of the design specification. Therefore, we provide two further page parameters, *sMin* and *sMax*, for the minimal and the maximal extension of the separating space.

A pagination assigns a page sequence number to each line and each figure, so it is an *onto* mapping

$$P{:}\{t_1,\ldots,t_m\} \cup \{f_1,\ldots,f_n\} \longrightarrow \{1,\ldots,p\},$$

where p is the total number of pages used. Of course, a pagination P must satisfy the following four basic constraints:

First, it must preserve the sequence of lines and figures; that is, if $t_i \leq t_j$ ($f_i \leq f_j$), then $P(t_i) \leq P(t_j)$ ($P(f_i) \leq P(f_j)$).

Second, the amount of material that can be placed on any single page is limited by the page parameters. The first constraint implies that for each page number q, $1 \leq q \leq p$, there are unique sequences $t_k,\ldots,t_l$ of lines and $f_u,\ldots,f_v$ that get assigned to page q by P. But page q must be neither overfull nor underfull, meaning that

$$\sum_{i=k}^{l} ht(t_i) + \sum_{i=k}^{l-1} htMin(\tau_i) + \sum_{i=u}^{v} ht(f_i) + \sum_{i=u}^{v-1} htMin(\phi_i) \Big[+ sMin \Big] \leq htMax$$

and

$$\sum_{i=k}^{l} ht(t_i) + \sum_{i=k}^{l-1} htMax(\tau_i) + \sum_{i=u}^{v} ht(f_i) + \sum_{i=u}^{v-1} htMax(\phi_i) \Big[+ sMax \Big] \geq htMin.$$

In these two equations, the bracketed terms *sMin* and *sMax* are only added if the page has a mixed content of text and figures; that is, if $l \geq k$ and $v \geq u$. The second equation only has to hold if $q < p$; that is, the last page may be only partly full.

As can be seen from the two equations, in a pagination P, white space in the text and figure streams contributes to a page only if it appears between two lines or two figures who fall onto the same page; that is, white space disappears at page boundaries.

Third, the white space that disappears at page boundaries has to be a legal page end (i. e. τ_l*.legalPageEnd* = *true*).

The fourth condition on pagination states that no figure can be placed on an earlier page than its citation. The exact statement depends on the nature of the document, namely whether it is printed on single-sided or double-sided pages.

For a single-sided document, only the top face of the paper sheets is used, so that the reader sees only one page at a time. For documents of this type, no figure may be positioned on a page strictly prior to its citation; that is, only those paginations P are admissible that satisfy

$$P(R(f)) \leq P(f),$$

for each figure f.

For a double-sided document, both faces of the paper sheet are used for printing, so that the reader sees a spread of two facing pages simultaneously, a left-hand, even-numbered, *verso* page and a right-hand, odd-numbered, *recto* page. The spread number $S(p)$ can be calculated from the page number p by

$$S(p) = (p \, \mathrm{DIV} \, 2) + 1,$$

so that the first spread consists of just the first page, the second spread consists of the second and the third page, and so on.

In a double-sided setting, no figure may be positioned on a spread strictly prior to its citation; that is, only those paginations P are admissible, that satisfy

$$S(P(R(f))) \leq S(P(f)),$$

for each figure f.

To summarize: Our page model is described by five parameters, namely the minimal and the maximal page height, the minimal and the maximal separation between the figure and the text area, and the sidedness of the document. A design specification for a document assigns values to these five parameters. A pagination routine works with a text stream, a figure stream, and a design specification as its input. It places lines and figures onto pages so that the inherent order of the two input streams and the citation relation is respected, each page by itself conforms to the design specification, and no figure can be placed on an earlier page (or page spread) than its citation.

3 MEASURING PAGINATIONS

Currently, automated pagination routines use a first-fit approach, placing a figure on the next page after its citation that has room for it. This greedy strategy is in conflict with the other goal of good pagination, to always fill each page according to the design specification. Figure 1 shows a thumbnail sketch of a document ([5], Chapter 3), which was paginated according to the first-fit strategy. An arrow originates from the citation of a figure and ends on the page where the figure is placed; text lines are omitted. As can be seen from Figure 1, the quality of the first-fit pagination is rather poor.

In consequence, page make-up specialists in book production rather resort to completely manual pagination than relying on first-fit automation. Figure 2 shows the result of this time-consuming process as applied to our sample document. Hand tuning in this case involved shrinking the two figures on pages 8 and 12 a bit so that they fit onto the same page as their citations and forcing the figure on page 13 onto an earlier page. Incidentally, this is how the document was actually printed. Our goal is to improve the automated process to a degree that only occasionally is manual intervention necessary.

Plass [2] was the first to research optimizing approaches to the pagination problem from an algorithmic point of view. His goal was to find paginations that place figures as closely as possible to their citations. He has suggested two goal functions, one linear and one quadratic, that measure paginations by the total number of pages that lay between

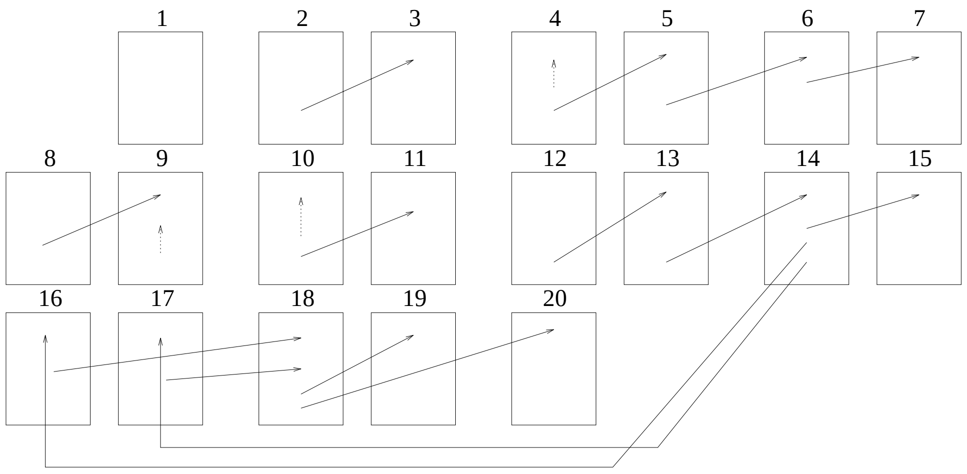

Figure 1. LaTeX*'s pagination* L_{100}

figures and their citations. In the notation of the previous section, the linear goal function for single-sided documents is

$$Lin(P) = \sum_{i=1}^{n} \Big(P(f_i) - P(R(f_i))\Big),$$

and the quadratic goal function is

$$Quad(P) = \sum_{i=1}^{n} \Big(P(f_i) - P(R(f_i))\Big)^2.$$

The pagination problem can then be posed as an optimization problem, namely to find a pagination P that minimizes a given goal function.

Plass argues that the quadratic goal function reflects better than the linear goal function the intuitive notion of quality that a human reader has of a pagination. We disagree, since the quadratic goal function overvalues small improvements of very bad placements. Consider a pagination P_1 that places 18 figures on the same page each as their citations and 1 figure 10 pages behind its citation. If measured with the quadratic goal function, this placement is considered worse than a pagination P_2 that places each of the 18 figures on the page after its citation and the last figure only 9 pages behind its citation:

$$Quad(P_1) = 18 \cdot 0^2 + 1 \cdot 10^2 = 100,$$

$$Quad(P_2) = 18 \cdot 1^2 + 1 \cdot 9^2 = 99.$$

However, we contend that a reader would rather have 18 figures ideally placed and not care whether the page difference for the last figure can be lowered from 10 to 9.

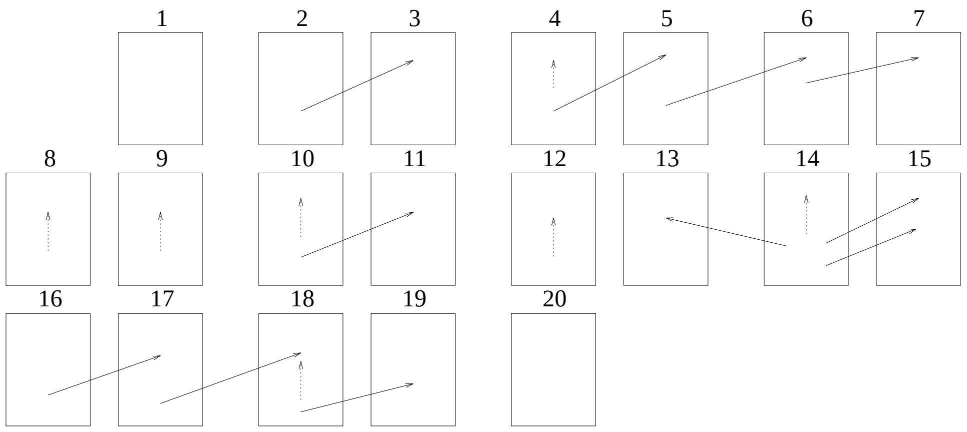

Figure 2. Hand-tuned pagination H_{100}

Therefore, we are not overly concerned with the time complexity of finding an optimal pagination with respect to the quadratic goal function. Plass has shown that this problem is NP-complete, but he has used a different framework: In his model, figure references may cross and figures may also be placed on pages before their citations.

We propose a linear goal function that minimizes the number of page turns a reader has to perform when reading the document from front to back. Furthermore, we introduce weights α for thumbing from a figure citation to the figure and back and β for the regular progression from page to page. Hence, our goal function is

$$Turn.S(\alpha,\beta,P) = \sum_{i=1}^{n} \alpha\Big(P(f_i) - P(R(f_i))\Big) + \beta(p-1).$$

Choosing $\alpha = 1$ and $\beta = 0$, we get Plass's original linear goal function. Choosing $\alpha = 0$ and $\beta = 1$, the prime concern is to produce as few pages as possible.

In the double-sided setting, we consider page spreads rather than single-sided pages. Therefore, as a second goal function, we propose to minimize

$$Turn.D(\alpha,\beta,P) = \sum_{i=1}^{n} \alpha\Big(S(P(f_i)) - S(P(R(f_i)))\Big) + \beta(S(p)-1).$$

Our algorithm *Turn.S*-Optimizer depends on the values of the parameters α and β as well as on the page specification (see Section 2), in particular on the page fullness *htMin*/*htMax*. Among all paginations that satisfy the page specification, our algorithm finds a pagination P that is optimal with respect to the goal function $Turn.S(\alpha,\beta,P)$.

We demonstrate in Section 5 with our sample document that paginations that are optimized with respect to page turns are superior to the results of the first-fit strategy. Even better results are achieved in the double-sided setting, optimizing spread turns with our algorithm *Turn.D*-Optimizer.

4 THE ALGORITHM

The dynamic programming approach [7] to pagination computes an optimal pagination $P_{i,j}$ for each subproblem $t_1, \ldots, t_i, f_1, \ldots, f_j$. The algorithm starts with $P_{0,0}$, a pagination that places nothing on zero pages. $P_{m,n}$ is then an optimal pagination of the whole document.

We represent $P_{i,j}$ as a record that contains at least three components: *paginationPossible* of type *boolean*, *noOfPages* of type *integer* and *predecessor* of type *integer* $\times$ *integer*.

```
funct ComputeOptimalPagination $\equiv$
    $P_{0,0} := (TRUE, 0, NIL \times NIL)$;
    for $j := 0$ to $n$ do
        for $i := 0$ to $m$ do
            if $(i = 0) \wedge (j = 0)$ then next fi
            $P_{i,j} := ($false$, -1, (NIL \times NIL))$;
            for $a := 0$ to $i$ do
                for $b := 0$ to $j$ do
                    if $FormsOnePage(a,b,i,j)$
                      then $P_{i,j}.paginationPossible :=$ true ;
                           $P_{i,j}.predecessor :=$
                             $BetterPredecessor(P_{i,j}.predecessor, (a \times b))$;
                           $P_{i,j}.noOfPages := P_{P_{i,j}.predecessor}.noOfPages + 1$;
                    fi
                od
            od
        od
    od
end
```

To compute $P_{i,j}$ the algorithm inspects all $P_{a,b}$ with $a \leq i$ and $b \leq j$ (but $(a,b) \neq (i,j)$!) and checks if $t_{a+1}, \ldots, t_i$ and $f_{b+1}, \ldots, f_j$ fit together onto *one* page. Of all possible $P_{a,b}$ that are valid predecessors of $P_{i,j}$, one that optimizes our goal function is chosen and stored as predecessor of $P_{i,j}$.

Notice that this algorithm needs time $O(m^2n^2)$. However the loops in *ComputePagination* can be shortened, leading to the time complexity $O(mn)$. This is due to the fact that pages usually have constant height. Then there is a maximum number of text lines and figures that fit onto one page.

FormsOnePage has to check if all the constraints are satisfied, that is

1. $P_{a,b}$ is a valid pagination.
2. The heights of $t_{a+1}, \ldots, t_i$ and $f_{b+1}, \ldots, f_j$ together with *sMin* and *sMax* fill one page according to the required minimum fill level.
3. The attribute *legalPageEnd* of τ_a is set.
4. No figure is placed before its reference ($R(f_b) \in \{t_1, \ldots, t_a\}$).

```
funct $FormsOnePage(a,b,i,j)$ $\equiv$
    if $(a = i) \wedge (b = j)$ then return false fi;
```

```
        if $\neg P_{a,b}$.paginationPossible then return false fi;
        if $\neg \tau_i$.legalPageEnd then return false fi;
        if $R(f_j) \notin \{t_1, \ldots, t_i\}$ then return false fi;
        if $(j - b) > 0 \wedge (i - a) > 0$
          then
                $hMin := sMin; hMax := sMax$;
          else
                $hMin := 0; hMax := 0$;
        fi
        $hMin := hMin + TextSum(a,i) + MinTextSpaceSum(a,i)$;
        $hMin := hMin + FigureSum(b,j) + MinFigureSpaceSum(b,j)$;
        $hMax := hMax + TextSum(a,i) + MaxTextSpaceSum(a,i)$;
        $hMax := hMax + FigureSum(b,j) + MaxFigureSpaceSum(b,j)$;
        if $hMin > pageHeight$ then return false fi;
        if $hMax < pageHeight$ then return false fi;
        return true ;
end
```

TextSum and *FigureSum* add up the heights of the text lines $t_{a+1} \ldots t_i$ and the figures $f_{b+1} \ldots, f_j$. *MinTextSpaceSum* adds up *htMin* of the spaces $\tau_{a+1} \ldots \tau_{i-1}$ in the text stream. *MaxFigureSpaceSum* adds up *htMax* of the spaces $\phi_{b+1} \ldots \phi_{j-1}$. If figures are allowed to appear not only on the top of a page but also on the bottom or in the middle, only *FormsOnePage* has to be changed.

The function *BetterPredecessor* compares two pagination candidates $P_{i,j}$*.predecessor* and $P_{a,b}$ and returns one that gives rise to a better pagination for $t_1, \ldots, t_i$ and $f_1, \ldots, f_j$. To compare two candidates, we have to measure the quality of a pagination $P_{i,j}$ for $t_1, \ldots, t_i$ and $f_1, \ldots, f_j$ even if there are dangling references; that is, if some figures f_k, $k > j$, are referenced by text lines in $\{t_1, \ldots, t_i\}$; that is, if $R(f_{j+1}) \leq t_i$. For such paginations, we have to generalize our goal function *Turn.S*. For a pagination $P_{i,j}$ that places $t_1, \ldots, t_i$ and $f_1, \ldots, f_j$ onto p pages, let

$$\begin{aligned} Turn.S(\alpha,\beta,P_{i,j}) &= \sum_{l=1}^{j} \alpha\Big(P_{i,j}(f_l) - P_{i,j}(R(f_l))\Big) \\ &\quad + \sum_{\substack{k=j+1 \\ R(f_k) \leq t_i}}^{n} \alpha\Big(p + 1 - P_{i,j}(R(f_k))\Big) \\ &\quad + \beta(p-1); \end{aligned}$$

that is, we pretend that all dangling figures f_k, $k > j$, which are referenced by a text line in $\{t_1, \ldots, t_i\}$, are placed on page $p + 1$.

If $P_{a,b}$ is the predecessor of $P_{i,j}$, then

$$\begin{aligned} Turn.S(\alpha,\beta,P_{i,j}) &= Turn.S(\alpha,\beta,P_{a,b}) \\ &\quad + \text{ the number of dangling figures } f_k,\ k > j,\ R(f_k) \leq i \\ &\quad + 1 \text{ (for the current page).} \end{aligned}$$

Hence, the value of an optimal pagination $P_{i,j}$ equals the minimum value for all possible predecessors $P_{a,b}$ ($a \leq i, b \leq j$), plus the number of dangling references, plus one (for the current page). This formula represents the optimality principle of the dynamic programming approach.

To extend this algorithm to double-sided documents one has to allow dangling figures on even-numbered pages. To compute the better predecessor, only those dangling references at odd-numbered pages are counted. It is even possible to balance pages on a spread if the height interval of each page is stored. *FormsOnePage* has to check, if the new page is odd numbered, that the height interval of the new page is a match to the height interval of the previous page.

5 PRACTICAL RESULTS

We have tested our implementation on Chapter 3 of a real-world document [5], which contains 18 figures. All page specifications used in the following examples have the same maximal page height and the same minimal/maximal separation between figures. First we consider paginations for page specifications that require a fill level of 100%. We indicate the minimal fill level of a pagination by an index to its name, for example P_{68} denotes a pagination P with all pages being at least 68% full.

LaTeX's pagination L_{100} is shown in Figure 1. Only 3 figures are placed on the same page as their citation, one figure is even 3 pages away from its citation, seven figures are not on the same page spread as their citation. It took several hours to hand tune the input, so that an acceptable pagination was achieved. Figure 2 shows the result. Still, three figures couldn't be placed on the same spread as their citation. One of them is even on the page spread before its citation, although this is considered bad style [3][4] and should be avoided. Measuring LaTeX's pagination L_{100} with our goal functions (with $\alpha = \beta = 1$) yields $Turn.D(1,1,L_{100}) = 7 + 10 = 17$ and $Turn.S(1,1,L_{100}) = 20 + 19 = 39$. For the hand-tuned pagination H_{100} (see Figure 2) we have $Turn.D(1,1,H_{100}) = 3 + 10 = 13$ and $Turn.S(1,1,H_{100}) = 11 + 19 = 30$.

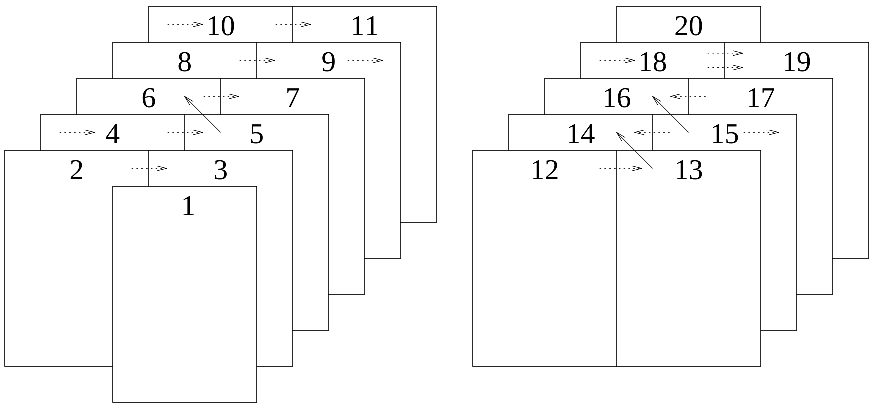

Figure 3. Turn.D$(1,1,\star_{100})$*-optimal pagination* D_{100} *with full double-sided pages*

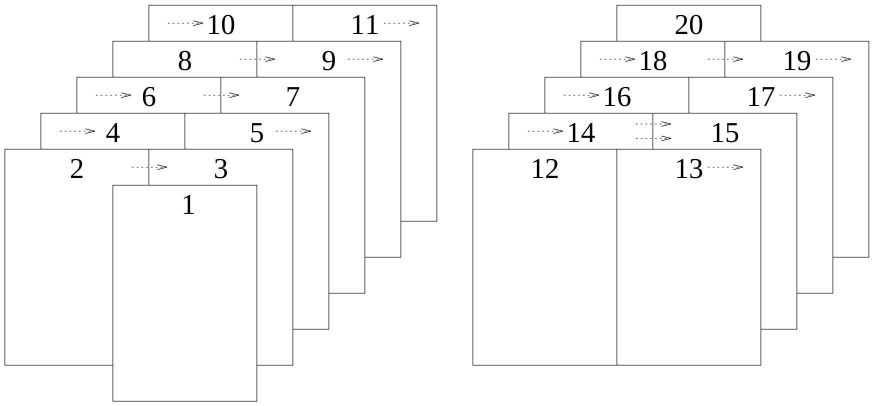

Figure 4. Turn.D$(1,1,\star_{90})$*-optimal pagination D_{90} with double-sided pages*

Figure 3 shows the pagination D_{100} computed by our algorithm. It is optimal with respect to our goal function *Turn.D* when pages have to be 100% full. We call this a *Turn.D*$(1,1,\star_{100})$-optimal pagination. Dotted arrows indicate a figure (at the end of the arrow) that is placed on the same spread/page as its citation (beginning of the arrow). Solid arrows indicate figures on different spreads than their citation.

Note that instead of 7 figures in LaTeX's pagination here only 3 figures are placed on a different spread as their citations. This is the same number of figures on different spreads as in the hand-tuned pagination. But the hand-tuned pagination has placed one figure on page 13 while the citation is placed on page 14. As has already been stated, this is considered bad style and should be avoided. In this case our algorithm performs a little better than a human. Measured with our goal functions *Turn.D*$(1,1,D_{100}) = 3 + 10 = 13$ and *Turn.S*$(1,1,D_{100}) = 13 + 19 = 32$.

It is still possible to improve the pagination even further, if pages need not be 100% full. The less full a page needs to be, the easier it is to place more figures on the same page as their citations, and the more pages are usually needed. Figure 4 shows that it is possible to place all figures on the same spread as their citation, if pages need only to be 90% full. This is a *Turn.D*$(1,1,\star_{90})$-optimal pagination with *Turn.D*$(1,1,D_{90}) = 0 + 10 = 10$ (and *Turn.S*$(1,1,D_{90}) = 6 + 19 = 25$). Of course we can't guarantee such good results for all documents. Just think of a document that only has three text lines that cite 20 large figures. But in such cases even human page make-up specialists can't find a good pagination.

The following table summarizes the results of the LaTeX pagination, the hand tuned pagination and the results of our algorithm.

pagination	*Turn.D*	*Turn.S*
LaTeX	$7 + 10 = 17$	$20 + 19 = 39$
hand-tuned	$3 + 10 = 13$	$11 + 19 = 30$
Turn.D$(1,1,\star_{100})$-optimal	$3 + 10 = 13$	$13 + 19 = 32$
Turn.D$(1,1,\star_{90})$-optimal	$0 + 10 = 10$	$6 + 19 = 25$

Next we look at the differences between the single-sided and the double-sided mode. The profit the algorithm gets from using double-sided pages becomes clear after a look

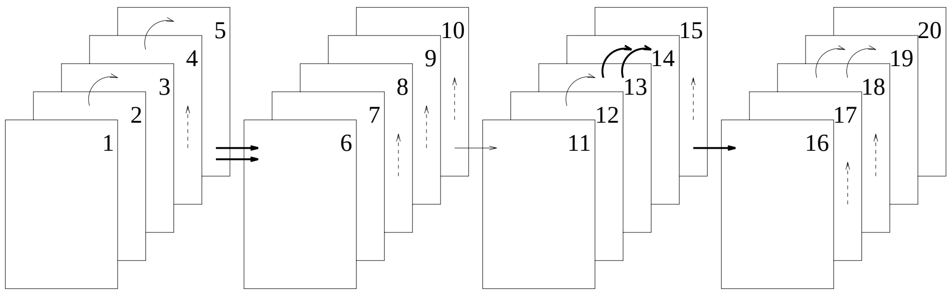

Figure 5. Turn.S$(1,1,\star_{100})$*-optimal pagination* S_{100}

at Figure 5. Here bold arrows indicate a figure not only on a different page but also on a different spread than its citation.

The optimal single-sided pagination S_{100} has two references from page 5 to page 6. On double-sided documents this is a spread boundary, so it is preferable to have only one reference from page 5 to page 6, as in D_{100} in Figure 3.

To get a single-sided pagination with all figures on the same page as their citation often requires the pages to be very loosely filled. In our example we had to decrease the minimum fill level to 70% to find such a pagination (see Figure 6). The result needed 23 pages instead of 20 pages with all pages exactly filled. But this pagination is also optimal with $Turn.S(1,1,S_{70}) = 0 + 23 = 23$ (and $Turn.D(1,1,S_{70}) = 0 + 12 = 12$).

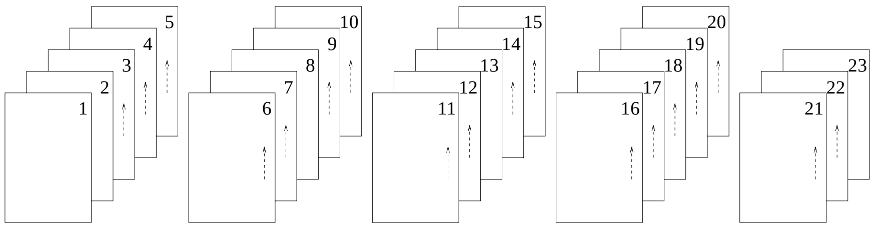

Figure 6. Turn.S$(1,1,\star_{70})$*-optimal pagination* S_{70}

The following table summarizes the results in double-sided and single-sided mode.

pagination	*Turn.D*	*Turn.S*
$Turn.D(1,1,\star_{100})$-optimal	$3 + 11 = 14$	$13 + 20 = 33$
$Turn.D(1,1,\star_{90})$-optimal	$0 + 11 = 11$	$6 + 20 = 26$
$Turn.S(1,1,\star_{100})$-optimal	$5 + 11 = 16$	$11 + 20 = 31$
$Turn.S(1,1,\star_{70})$-optimal	$0 + 12 = 12$	$0 + 23 = 23$

The example shows, that the optimal result for single-sided pages is not optimal, if double-sided pages are available and vice versa.

The difference between our goal function *Turn.S* and the linear goal function *Lin* is demonstrated by Figure 7, which shows a *Lin*-optimal pagination with a minimum fill level

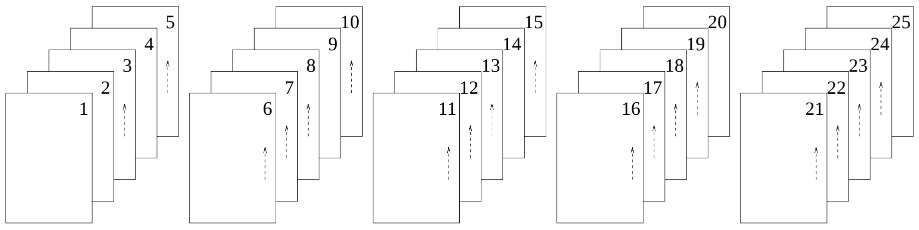

Figure 7. Optimal linear pagination with min. 70% filled single-sided pages

of 70%. The algorithm that only aims to optimize *Lin* is at liberty to use 25 pages instead of the 23 that *Turn.S*-Optimizer uses for the same fill level of 70% and for a perfect pagination that places each figure onto the same page as its reference. *Turn.S*, by taking into account the number of pages, too, and hence differentiating where *Lin* identifies, reflects better the user's judgement on the merits and demerits of a pagination than *Lin* does.

The implementation of our algorithm is written in C++, but it is a first prototype that is neither optimized for speed nor storage. The time to compute the pagination of our sample document was about one minute on a SUN Sparc 20.

6 CONCLUSIONS AND FURTHER WORK

We have presented an algorithm and a goal function for the pagination of documents. Our approach is better than the algorithms used in today's formatters. It seems that it is even better than a page make-up specialist doing pagination by hand. In order to validate this assumption more documents need to be formatted with our approach.

Asher has implemented an optimizing pagination routine into the publishing system Type & Set [8]. His goal function, although not explicitly presented, seems to be mainly concerned with page justification and balancing, and not so much with figure placement. Our algorithm is capable also of handling the justification of pages and the balancing of columns across a page or a page spread, as well as of freezing parts of the previous pagination when reformatting after updates.

So far our implementation only deals with two input streams, namely text and figures. But usually there are more floating objects than figures. In fact some of the "figures" in our example document were tables. We restricted a table *i*, cited between figures *j* and *k* to be placed between these figures. The style guides also allow the table to be placed before or after these figures. It is only necessary to place all tables in order of their citation and all figures in order of their citations. This may increase the quality of the pagination. Our approach works with more input streams, but this has not been implemented yet.

A special type of floating object is the footnote. Footnotes are cited in the text, but unlike figures they must start on the same page as the citation. If the last footnote on a page doesn't fit completely on the page, the footnote may be continued on the next page. We plan to optimize pagination for documents with figures and footnotes next.

We do not foresee that automated pagination routines, even when optimizing, can turn out acceptable paginations for all documents. Difficult cases will always require human intervention. Methods of interaction are needed that enable make-up specialists to explicitly place figures on specific pages and in general to strengthen or to relax constraints locally, for example constraints on the fill level of pages. Our next goal is to provide support for manual interaction with pagination routines.

Our overall strategy is different from the one employed by Kernighan and Van Wyk in their troff postprocessor PM [9]. Kernighan and Van Wyk strive to make the pagination routine as simple and efficient as possible, using basically a first-fit approach; they rely then on the author or typist to move figures around in the input text when their placement needs improving. The strong point of this approach is that the freezing of the pagination for as long a prefix of the document as has not changed, is guaranteed. In contrast, with optimizing approaches, freezing up to a specific point in the input has to be explicitly requested.

Our aim is to have the pagination routine automatically produce results that are so good that the need for manual intervention is minimized.

Groves and Brailsford [10] base their first-fit pagination algorithm on the block model of Kernighan and Van Wyk. The primary concern of their algorithm is the balancing of pages and the avoidance of orphans and widows. Blocks that can be broken across pages but have a minimal “need” are used to achieve the latter. In our model, we set the *legalPageEnd* attributes of the first and the last-before-last lines of a paragraph to *false* to achieve the same goal. Indeed, it seems both possible and attractive to re-implement our optimizing algorithms for the block model as an extension of Groves and Brailsford’s *dlink* system.

ACKNOWLEDGEMENTS

This work was supported by the Deutsche Forschungsgemeinschaft, grant BR 1309/2-1.

REFERENCES

1. Frame Technology Corporation, San Jose, California, *FrameMaker User Manual*, 1995.
2. M. F. Plass, ‘Optimal pagination techniques for automatic typesetting systems’, Technical Report STAN-CS-81-870, Department of Computer Science, Stanford University, (1981).
3. H. Williamson, *Methods of Book Design*, Yale University Press, New Haven, 1983.
4. The Chicago manual of style. The University of Chicago Press, Chicago, 1982.
5. Wissenschaftliche Information im elektronischen Zeitalter. Stand und Erfordernisse. Bayerisches Staatsministerium für Unterricht, Kultus, Wissenschaft und Kunst, RB-05/95/14. Available on the WWW by the URL http://www11.informatik.tu-muenchen.de/EFI/, July 1995. Bericht der Sachverständigenkommission zur elektronischen Fachinformation (EFI) an den Hochschulen in Bayern.
6. Donald E. Knuth, *The TEXbook*, volume A of *Computer & Typesetting*, Addison Wesley Publishing Company, Reading, MA, 1986.
7. Thomas H. Corman, Charles E. Leierson, and Ronald L. Rivest, *Introduction to Algorithms*, McGraw-Hill Book Company, 1990.
8. G. Asher, ‘Type & Set: TEX as the engine of a friendly publishing system’, in *TEX: Applications, Uses, Methods*, ed., M. Clark, pp. 91–100, Chichester, UK, (1990). Ellis Horwood Publishers. Proceedings of the TEX88 Conference.
9. B. W. Kernighan and Ch. J. Van Wyk, ‘Page makeup by postprocessing text formatter output’, *Computing Systems*, **2**(2), 103–132, (1989).
10. Michael J. Groves and David F. Brailsford, ‘Separate compilation of structured documents’, *Electronic Publishing—Origination, Dissemination, and Design*, **6**(4), 315–326, (December 1993).