Journal of Heuristics, 2: 321–342 (1997)
© 1997 Kluwer Academic Publishers

Automatic Yellow-Pages Pagination and Layout

RAMESH JOHARI
Harvard University

JOE MARKS
MERL

ALI PARTOVI
mFactory

STUART SHIEBER
Harvard University

Abstract

The compact and harmonious layout of ads and text is a fundamental and costly step in the production of commercial telephone directories ("Yellow Pages"). We formulate a canonical version of Yellow-Pages pagination and layout (YPPL) as an optimization problem in which the task is to position ads and text-stream segments on sequential pages so as to minimize total page length and maximize certain layout aesthetics, subject to constraints derived from page-format requirements and positional relations between ads and text. We present a heuristic-search approach to the YPPL problem. Our algorithm has been applied to a sample of real telephone-directory data, and produces solutions that are significantly shorter and better than the published ones.

Key Words: directory pagination, page layout, heuristic search, stochastic optimization, simulated annealing

1. Introduction

The problem of Yellow-Pages (YP) pagination and layout (YPPL) is one of finding a compact and harmonious positioning of text and advertisements on the pages of a commercial telephone directory.

Figure 1 depicts a sample page from a commercial Yellow Pages directory. We use this figure to define some useful terminology. Regular *ads*, which can span more than one column horizontally, are arranged on the page in a packing. The *text stream* giving listings in alphabetic order (including *in-line ads* that are considered part of the text stream) is used to fill in the remaining space. Any space remaining after the ads and text are placed is filled with *filler* material, and is essentially wasted space. The filler material can be further subdivided into *bubbles*, space trapped underneath an ad, and *padding*, space added at the end of a text column to adjust the assignment of text to columns and pages.

The compactness criterion derives directly from cost: fewer pages make for lower printing costs. Since the ad and text streams are fixed, compactness is achieved by reducing the amount of filler. A typical YP directory in the United States contains approximately 10%

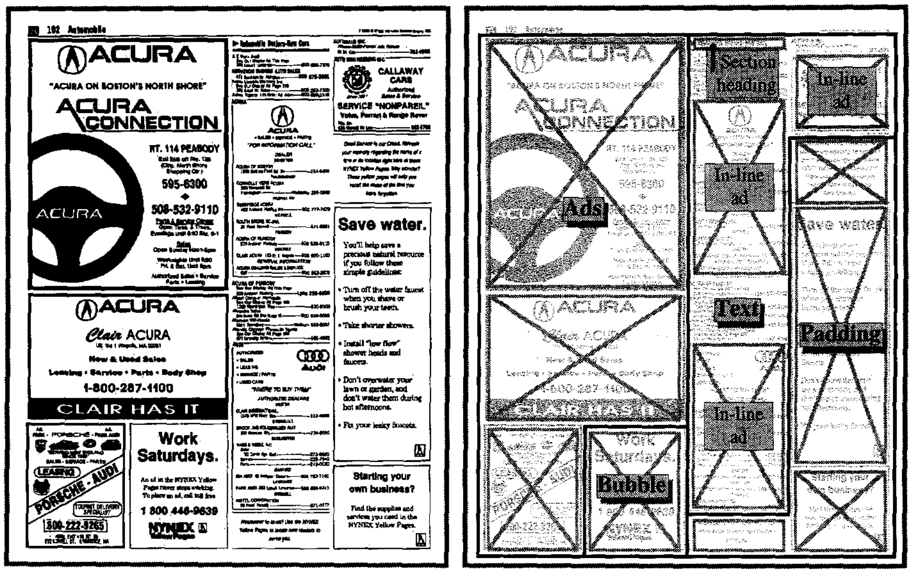


Figure 1. A sample page from the NYNEX Yellow Pages.

wasted space. With 350,000,000 commercial directories printed annually in the United States alone (Yellow Pages Publishers Association (1995)), the wastage is considerable. Thus the achievement of more compact layouts without sacrificing layout quality could be very useful. Of course, improving layout quality at the same time would be even better.

Layout quality is characterized by a notion of harmonious placement, which comprises various layout criteria and conventions relating to page-format requirements and text-ad relationships. These criteria and conventions vary from country to country and publisher to publisher. In this paper, we consider an idealization of the version of YPPL used by most YP publishers in the United States, though the method we propose, because it is based on a weak stochastic optimization algorithm, is applicable to widely different notions of layout quality. In the present version, ads and text are laid out in a four-column format, with the text positioned above the ads in each column. Each regular ad is required to be in the same section as its corresponding entry in the text stream, that is, on the same page as the start or end of the appropriate section, or on some page in between. Smaller ads cannot appear on an earlier page than larger ads within a section. Furthermore, it is desirable to have ads as close to the start page of a section as possible. Finally, the text stream should not be broken infelicitously (for example, placing a column break immediately after a section heading).

1.1. *Previous work*

Document-formatting problems arise in the production of all kinds of published material. Early research in this area focused on text formatting (Peels, Janssen, and Nawjin (1985), Furuta, Schofield, and Shaw (1982), Knuth and Plass (1981)). True pagination and page-layout problems have received less attention, though the eclectic nature of the little work that has been done (Plass (1981), Iwai et al. (1989), Rosenking et al. (1991), Chew et al. (1994), Weitzman and Wittenburg (1994), Harada, Witkin, and Baraff (1995), Graf, Neurohr, and Goebel (1996)) illustrates the broad range of problems that arise and the solutions that might be applied to them.

Of previously reported approaches, the three most relevant for general pagination and page-layout tasks are those that use dynamic programming, rule-based methods, and constraint-based methods.

The YPPL problem is an inherently difficult one, as it involves two variants of the bin packing problem (initially for ad placement and then for text-stream breaking in the resulting available space). Of course, some quite similar variants of bin packing are efficiently solvable by dynamic programming. For instance, the optimal line-breaking problem, which bears close resemblance to the text-stream breaking problem, is solvable in linear time by a dynamic programming algorithm (Knuth and Plass (1981)). However, since text-stream breaking relies on the configuration of ads, and ad placement relies on the positioning of text (in particular, on the placement of the section headers), the two must be solved together; any dynamic-programming solution must handle ads as well as text. Since certain simple variants of the ad-placement problem are trivially NP-hard (by reduction from bin-packing) no such polynomial-time solution is likely to exist for these variants.

However, certain restricted variants of YPPL, namely those in which ads must be retained in a specific order, may be amenable to dynamic programming solutions along the lines of the page-layout algorithms of Plass (1981). There are two disadvantages with pursuing this possibility. First, though the resulting algorithms are polynomial, their degrees would be prohibitive to use in practice. Second, variation among YPPL problems in their particulars is large; any given dynamic-programming solution would be for a particular highly constrained variant of the problem, whereas the heuristic methods we propose here are much more general in application.

The rule-based approach has been applied to the YPPL problem and its newspaper equivalent, apparently with moderate success (Chew et al. (1994), Rosenking et al. (1991)). While rule-based systems may be quite useful in an assisting role, and "if-then" rules may even afford a convenient way of defining an objective function operationally, it seems unlikely that they are suitable for full automation of what is a very challenging combinatorial-optimization problem. Rule-based approaches to cartographic label placement (another computationally challenging graphical layout problem) were also once considered to be very promising, but now appear to have been superseded by other forms of heuristic search and optimization (Zoraster (1991), Christensen, Marks, and Shieber (1995)).

Graf et al. have recently described the use of constraint-based methods for YPPL (Graf, Neurohr, and Goebel (1996)). An implemented system based on their technology is being tested at several sites, but no assessment of its effectiveness has been reported yet.

1.2. Overview of approach

Our approach to automatic YPPL uses a heuristic search technique, simulated annealing (Kirkpatrick, Gelatt, Jr., and Vecchi (1983), Černy (1985)). The naive application of stochastic techniques like simulated annealing to complex and intractable problems is often fruitless; this is true of the YPPL problem as well. To use stochastic search effectively, one often needs to craft an appropriate representation for candidate solutions, and a set of effective perturbation operators that transform one candidate solution into a closely related "neighboring" solution. The main contribution of this paper is a candidate-solution representation and a set of perturbation operators that enable simulated annealing (and possibly other heuristic methods) to be used effectively for YPPL. We have tested our algorithm by applying it to samples of actual YP data.

We begin by describing the problem and its variants in a more precise manner (Section 2). We then describe our algorithm in detail (Section 3). After a presentation of empirical results (Section 4), we conclude with a discussion of the capabilities and limitations of our current approach (Section 5).

2. Problem definition

The YPPL problem is a combinatorial optimization problem. In this section, we describe the space of problem instances, the space of solutions for a given instance, and various objective functions that might be optimized over the solution space.

A problem instance is defined by the text and ad streams and the page format:

- *Text stream*: An appropriately alphabetically ordered list of text entries (including section headers), each with a specification of its height and whether or not it constitutes a section header. (Width of entries is uniformly one column, thus requiring no overt specification.) For present purposes, in-line ads can be thought of as text entries with unusually large heights.
- *Ad stream*: A specification of the height, width, and matching text entry for each ad. (The section that an ad is associated with is determined by the section within which the matching text entry falls.)
- *Page format*: The page size, given by number of columns and column height.

A solution to such a problem instance comprises the following data:

- *Text breaks*: A specification of the entries before which column breaks occur. (Since some columns may have no text in them, multiple column breaks may precede a given text entry).
- *Ad placement*: A specification of the page and row and column position of each ad.

Of course, many such "solutions" will be ill-formed. For instance, ads or text may overlap or protrude off the page. Furthermore, even if no such problems occur, the ads may be placed poorly, far away from the section in which the ad should occur. In summary, the space of solutions characterized in this way includes a vast number of poor solutions. Nonetheless, this is an especially simple and direct characterization of the solution space. We leave it to the objective function to quantify the quality of a solution. Important components of an objective function include: syntactic considerations, such as elimination of overlapping ads

Table 1. General criteria for evaluating solutions.

Term	Description
Syntactic criteria	
OVERFLOW	Ads and text should not overflow the page boundaries
OVERLAP	Ads and text should not overlap each other
Compaction criteria	
NUM_PAGES	Number of pages should be minimized
Layout quality criteria	
OUT_OF_SECTION	Ads should be placed within their associated section, i.e., on the same page as the start or end of the section or on some page in between
DISTANCE_FROM_SECTION_START	Ads should be placed as close to their section header as possible
BAD_TEXT_BREAK	Text columns should not break immediately after section headings
ADS_OUT_OF_SIZE_ORDER	Ads should be in size order
UNEVEN_PADDING	The amounts of padding at the end of each text column should be approximately equal

or overflowing text, required to make the solution well-formed; compactness considerations, so that shorter layouts are preferred over longer layouts; and layout quality, so that ads are placed within and near the beginning of their sections, column breaks are reasonable, and so forth.

The particular objective function that we use here is an idealization of the version of YPPL used by most YP publishers in the United States. In this scheme, the ads must appear within the pages of the section that they are associated with, preferably as close to the beginning of the section as possible, and ads within a section must be ordered by size from largest to smallest (that is, a smaller ad must not appear on an earlier page than a larger one). A precise formulation of the objective function as a weighted sum of costs to be minimized is deferred to the next section, but the evaluative criteria are described in Table 1. For example, syntactic well-formedness requires that ads not overflow the boundaries of a page. Compactness prefers layouts with fewer pages. Finally, layout quality is promoted by costs that, for example, penalize ads out of their section or out of size order.

Given a complete problem instance, the YPPL problem is to position the ads and text-stream segments on sequential pages so that the page-format requirements are met and the objective function is minimized. In practice, large YPPL problems must be broken up into smaller subproblems, which are solved independently and whose results are concatenated to give a final complete solution. For instance, a 1000-page YP directory might be laid out using an “overlapping window” technique. Material comprising roughly the first 50 pages would be laid out, with the final page likely being only partially filled. The ads and text on this final page plus the next 50 pages or so of material make up the next window to be laid out, and so forth. Thus, about 20 runs of 50 pages each would lay out the entire directory. An alternative (but more costly) strategy would be to increase the overlap between the windows using the material from the last several pages in the subsequent window. For either decomposition strategy to work well, it is important that the subproblems be as large as possible so that full advantage can be taken of all the combinatorial possibilities for each subproblem, and it is essential that the subproblems be solved well. The near-optimal solution of 40- to 50-page instances of the YPPL problem is therefore the focus of the current paper.

3. Algorithm description

The key to an effective heuristic search method for the YPPL problem is to find a method for representing solutions indirectly, such that poor solutions are never considered at all. By contrast, imagine a direct implementation of a heuristic search method that generated solutions as text break and ad-placement specifications, using some stochastic search method such as random descent, simulated annealing, a genetic algorithm, etc. Search within this space would likely be hopeless, as most solutions are ill-formed and searching neighbors of such poor solutions provides insufficient direction toward finding even a plausible, let alone high-quality, solution.

The conclusion to draw, however, is not that stochastic search is inappropriate for the problem, but rather that the representation of the search space is inadequate. We must find a representation for candidate solutions that uses problem-specific information to facilitate searching. The most important aspect of our research is just such a representation scheme. We propose a representation for candidate solutions, which can be decoded into full text break and ad-placement information, but which does not specify this information literally.

3.1. Candidate-solution representation

We start by sorting the ads by section, and within each section by ad size. A candidate solution then comprises the following data:

- *Ad-stream page breaks*: Indices into the sorted ad stream that indicate where page breaks occur. From this, an ordered list of ads can be computed for each page. This therefore indirectly provides page information for the ads.
- *Ad columns*: The column positions of the ads on their respective pages; for ads that span multiple columns, the assigned column is conventionally taken to be the leftmost one.
- *Padding allotments*: Minimum amounts of padding (filler at the end of a text column) for each page. The addition of extra padding through the padding allotments provides a method for the representation to control at which pages sections start and end.

This representation leaves two major aspects of a complete pagination and layout solution unspecified. First, although ads are assigned to a specific page and column (or columns) in this representation, the row positions on their respective pages are not given in a candidate solution. Second, neither pagination nor layout of the text is specified explicitly. Deriving

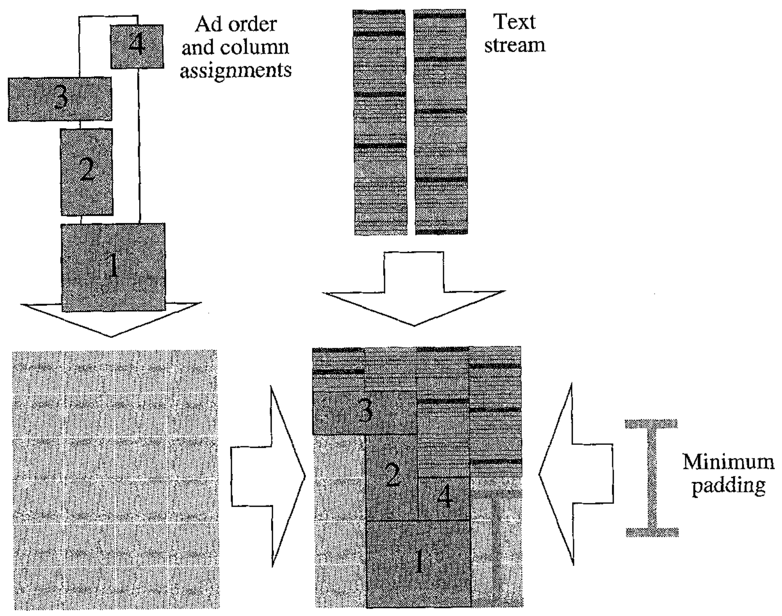


Figure 2.

a complete pagination and layout from a candidate solution requires a decoding process. This decoding process entails the following steps (see figure 2)[1]:

1. *Packing*: The ads assigned to each page are placed in order, by moving each ad as far down as possible in its column(s) without causing overlap before the next ad is considered. In the figure, ads 1 and two assigned to column 2, ad 3 is assigned to column 1, and ad 4 is assigned to column 3. When they drop into their respective columns in order, the ads stack as shown in the figure.
2. *Text-stream routing*: The text stream is routed through the space above the ads in each column, placing as much text into each column as will fit without requiring a break in the text stream after a section header. This process stops for each page before the ad areas, bubble areas, padding allotment, and text area together exceed the page capacity. In the figure, some padding is left at the end of column 2 to ensure that the column does not end with a section header. A larger amount of padding is left at the end of column 4 to ensure that the page meets its minimum-padding requirement.

Once the decoding process has been applied to a candidate solution, the resulting pagination and layout can be evaluated using the objective function in Table 2. This table presents the actual terms and weights used to implement the general criteria of Table 1.

This representation together with its decoding process has the property that it is impossible to represent a solution that includes overlapping ads, because the packing step of the decoder positions the ads in a nonoverlapping manner. Similarly, ads must occur in size order, as

Table 2. An objective function for the canonical version of YPPL. Indented terms replace or augment the terms immediately above them.

Term	Description	Value
Syntactic terms		
OVERFLOW	A fixed cost this is assessed once if any page has insufficient room for its assigned ads	50,000.0
— OVERFLOW_AREA	A cost per unit area of overflowing ads	200.0 × area
OVERLAP	NOT NEEDED	
Compaction terms		
NUM_PAGES	SUPERSEDED BY NEXT TWO TERMS	
— BUBBLE	A cost proportional to the total bubble area	6.0 × area
— PADDING	A cost proportional to the total amount of padding	3.0 × area
Layout quality terms		
OUT_OF_SECTION	A fixed cost for each ad that is not in the same section as its associated text entry	75,000.0 × # of ads
— DISTANCE_OUT_OF_SECTION	A cost proportional to the number of pages an out-of-section ad is away from the closest page of its section	500.0 × # of pages
DISTANCE_FROM_SECTION_START	A cost assessed for each ad that is in its section, but that is not on the first page of that section. The cost is proportional to the number of pages the ad is away from the section header	200.0 × # of pages
BAD_TEXT_BREAK	NOT NEEDED	
ADS_OUT_OF_SIZE_ORDER	NOT NEEDED	
UNEVEN_PADDING	A cost proportional to the amount by which the padding for each column on a page differs from the average amount of padding per column for that page	0.3 × area

they are initially sorted that way. The decoder also guarantees that text fits within the available area above the ads, and that columns do not end in section headers. Thus, the cost terms OVERLAP, ADS_OUT_OF_SIZE_ORDER, and BAD_TEXT_BREAK are guaranteed to be optimized and are not needed in the objective function; the search process does not need to search through solutions failing these most important criteria.

On the other hand, solutions with ad overflow or ads out of section are still representable; the search process is used in part to reduce such overflows. Therefore, the terms OVERFLOW and OUT_OF_SECTION must still be included in the objective function in order to strongly disfavor very undesirable candidate solutions. But given their inclusion, the two additional terms OVERFLOW_AREA and DISTANCE_OUT_OF_SECTION introduced in Table 2 may appear redundant. However, without these proportional terms, layouts with any overflow or out-of-section ads would be considered equivalent, regardless of the degree of the infraction. The lack of a quantitative distinction between layouts with a little overflow and those with a lot (respectively, between layouts with few ads out of section and those with many) would reduce the continuity of the search space and would therefore make for an unnecessarily difficult optimization task. For this reason, these two terms are added to aid in optimization of the more primitive OVERFLOW and OUT_OF_SECTION terms.

The final way in which the specific terms of Table 2 vary from the general criteria in Table 1 is in the implementation of the NUM_PAGES criterion. Rather than penalizing filler merely by adding a cost proportional to the number of pages used in the layout, we treat bubble and padding costs separately. Bubbles have a more dramatic effect on overall compactness of a layout as they cannot be varied directly and uniformly, since they arise as a side effect of the overall ad placement. Padding, on the other hand, is directly controllable in the representation. Consequently, we penalize bubble area more than padding area. As bubbles and padding are the only types of filler, minimizing these two minimizes pages.

A final note is pertinent on treatment of wasted space on the last page of a layout. Since the material that ends up on on the last page of a layout will be reconsidered as the starting material for the next window of material, the layout of this material will not be used, and its compactness is of no import to the eventual overall layout quality. (A similar argument holds for the layout of the last page on the last window.) Consequently, there is no reason to penalize wasted space on the final page of a layout, and the notion of "area" used in the computation of the BUBBLE and PADDING criteria excludes the area of bubbles and padding on the final page.

3.2. *Search algorithm*

Given a way of representing candidate solutions, we can use a stochastic search method such as random descent, simulated annealing, or a genetic algorithm to navigate the space in search of good solutions. All these methods operate by starting with an initial solution (or solutions) and then perturbing or recombining the solutions to generate new ones. The particular search method we used was a variant of simulated annealing (Kirkpatrick, Gelatt, Jr., and Vecchi (1983), Černy (1985)). Starting with an initial solution, the solution is perturbed and the generated solution is kept if better than the previous solution, and kept with diminishing probability if worse. When convergence is reached, the process is terminated. Figure 3 contains a pseudocode description of our particular variant of simulated annealing.

The search starts with the initial candidate solution described in Table 3. To generate neighboring solutions we use the perturbation operators described in Table 4. In addition to the SHIFT operator, we tried combinations of several other operations for modifying the ad-stream page breaks. These operators included:

- JOIN: Delete a randomly chosen page break.
- SPLIT: Add a page break at a randomly chosen location in the ad stream.

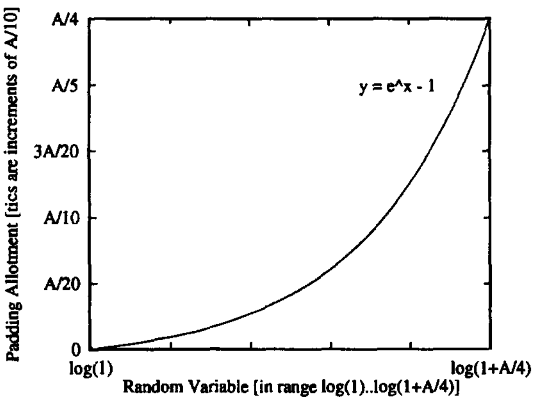
```
Let current be the initial candidate solution
best ← current
t ← StartTemp
numAccept ← 0
consecFrozen ← 0
While (consecFrozen < FrozenLimit) do
{
    i ← 0
    While (i < TempLength) and (numAccept < MaxAccept) do
    {
        Perform a perturbation on current, producing new
        Decode and evaluate new
        If (Cost(new) < Cost(current))
        {
            Accept perturbation: current ← new
            numAccept ← numAccept + 1
            If (Cost(current) < Cost(best))
                best ← current
        }
        else with probability $e^{(\mathrm{Cost}(current)-\mathrm{Cost}(new))/t}$
        {
            Accept perturbation anyway: current ← new
            numAccept ← numAccept + 1
        }
        i ← i + 1
    }
    If (numAccept < MinAccept)
        consecFrozen ← consecFrozen + 1
    else
        consecFrozen ← 0
    t ← TempFactor · t
}
Return best


Parameter values (N is the number of ads):
    StartTemp = 700.0              FrozenLimit = 5
    TempLength = 100N              MaxAccept = 35N
    MinAccept = 6                  TempFactor = 0.97
```

Figure 3. Pseudocode for a variant of the simulated-annealing search technique.

Table 3. The initial candidate solution.

Data	Initial values
Page ads	The initial number of pages is chosen to be the number needed to hold all the text and ads, plus 2%–6%. Ads are evenly distributed over these pages according to area, while maintaining the overall ad-stream order
Ad columns	Uniform random choice
Padding allotments	0.0 for every page

Table 4. Perturbation operators.

Operator	Description	Probability
SHIFT	Remove a randomly chosen ad-stream page break, and add another one at a randomly chosen slot in the ad stream	0.4
XMOVE	Randomly set the column position for a randomly chosen ad	0.3
PAD	Set the padding allotment of a randomly chosen page to a value $y = e^x - 1$, where x is uniformly distributed in the range $[\log(1), \log(1 + \frac{A}{4}]$, with A being the page area A graph of this distribution is as follows:  The effect is that padding allotments are themselves chosen from the range $[0, A/4]$, that is, less than one column, with preference for smaller allotments	0.3

- SHRINK: Decrement by one the places in the ad stream of a randomly chosen page break and all subsequent page breaks.
- GROW: Increment by one the places in the ad stream of a randomly chosen page break and all subsequent] breaks.

However, lengthy experimentation led us to the conclusion that the SHIFT operator alone was most effective.

4. Empirical tests

4.1. Results data

We tested the algorithm described above using 46.25 pages' worth of data collected from a published commercial telephone directory[2]. The results reported here are from the first and only tests on this data. A separate data set was used to develop and tune our algorithm, to avoid the problem of tuning our algorithm to the test data set. The results for the development data set were slightly better than those given here, and are summarized below in footnotes. Excluding filler, the ads and text in our test problem take up 40.8 pages. Allowing 2%–6% for filler gives a range of 42–44 pages, so we ran three sets of experiments in which ads were distributed initially over 42, 43, and 44 pages, respectively (see Table 3)[3].

Figures 4 through 6 plot objective-function scores (using the objective function from Table 2) against the number of iterations (one candidate solution is formulated and evaluated per iteration) for 20 runs of the algorithm, all seeded with different random numbers. (One million iterations of the algorithm on this data set requires approximately two hours on a DEC 3000/400 AXP workstation.) To render the objective-function scores more meaningful,

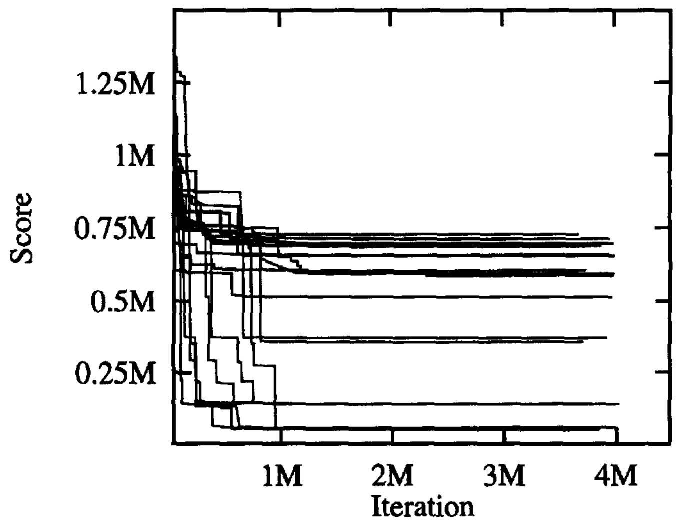


Figure 4. Cost curves for 42-page initial ad distribution.

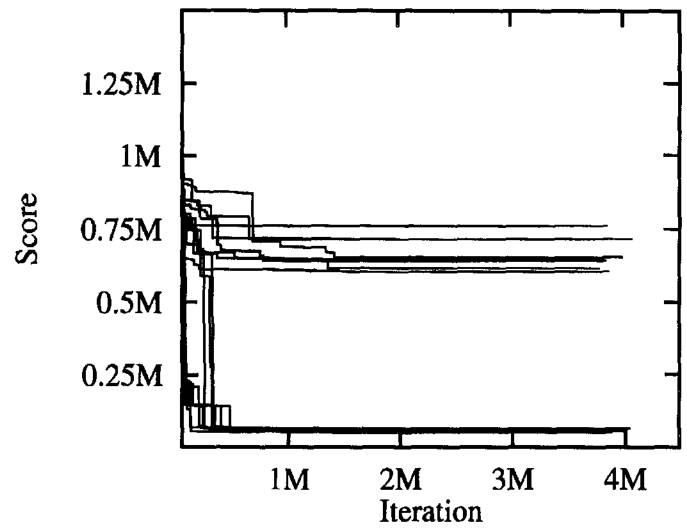


Figure 5. Cost curves for 43-page initial ad distribution.

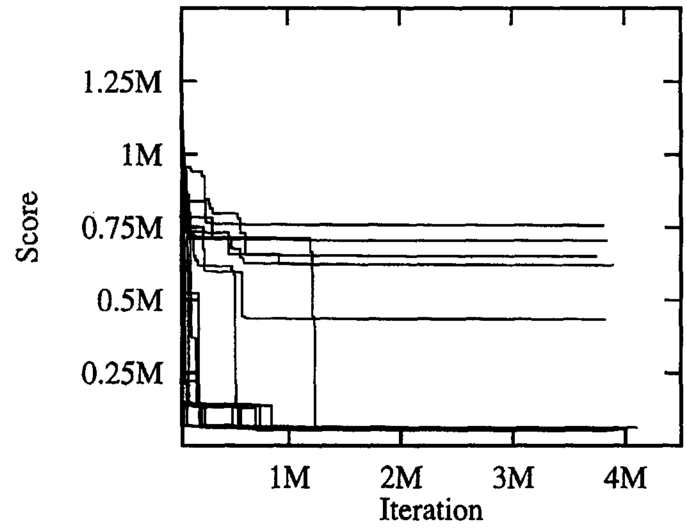


Figure 6. Cost curves for 44-page initial ad distribution.

Tables 5 through 7 show how the raw scores relate to the three most relevant measures of pagination and layout worth: the number of pages required (for simplicity the last page is counted, even if it is not full); the number of ads that were placed out of their respective sections; and whether or not any pages had overflowing ads. The comparable values from the telephone directory are 47 pages, zero ads out of section, and, of course, no page overflow[4].

Table 5. Partial score breakdown of 20 runs with 42-page initial ad distribution.

Score	# of pages	# of ads out of section	No page overflow
49376	42	0	√
50582	43	0	√
55983	43	0	√
57011	42	0	√
137647	43	1	√
138014	43	1	√
138875	43	1	√
356582	43	4	√
369714	43	4	√
512462	44	6	√
585706	42	0	
592960	43	0	
604066	42	0	
653427	43	0	
656486	43	0	
683515	43	0	
687246	43	0	
695364	42	0	
710310	42	0	
726085	43	0	

The results in Tables 5 through 7 suggest that distributing the ads over 42 pages initially is too optimistic. Although this gives the two best layout scores, the "yield", or number of satisfactory solutions, is higher for both the 43-page and 44-page distributions: 60% and 70% of the solutions, respectively, satisfy all the constraints and are more compact than the published directory solution. This contrasts with a 20% yield for the 42-page distribution[5]. We therefore recommend distributing the ads initially over the minimum number of pages required plus 4%, which in this case rounds up to 43 pages[6].

Finally, the best solution of the 20 with a 43-page initial ad distribution (the distribution we recommend) is shown in figures 7 through 10: ads are drawn in white with crossbars; filler is drawn in lightest gray; regular text entries are in medium gray; section headers are in darkest gray; and the numbers indicate the correspondences between ads and section headers. The distribution of ads with respect to their section starts for this solution and for the directory solution is tabulated in Table 8[7].

Even though this solution improves greatly on the published directory solution, manual inspection can still yield small gains. For example, the average distance to section start for

Table 6. Partial score breakdown of 20 runs with 43-page initial ad distribution.

Score	# of pages	# of ads out of section	No page overflow
50736	42	0	✓
53130	43	0	✓
56503	43	0	✓
56882	43	0	✓
59527	43	0	✓
59576	43	0	✓
61145	43	0	✓
62945	43	0	✓
63290	43	0	✓
64231	43	0	✓
64486	43	0	✓
64530	43	0	✓
604099	43	0	
615107	42	0	
641388	43	0	
646799	42	0	
653268	43	0	
715274	42	0	
715609	42	0	
760494	42	0	

ads in Section 1 can be decreased slightly by a multistage transfer comprising the following moves: one 3×1 ad from Page 4 to Page 3; one 2×1 ad from Page 5 to Page 4; and two 1×1 ads from Page 6 to Page 5. A more likely reason to adjust computed solutions would be to improve the arrangement of text and ads *within* a page. For example, the contents of Page 17 might be rearranged to better synchronize the ads and text. Although better intrapage layout aesthetics could be incorporated into the decoding process described in Section 3.1, minor adjustments of the kind just mentioned may also be done easily by manual inspection.

4.2. Alternative methods

A variety of representations were devised and tested for the YPPL problem before arriving at the one described above. In Section 3.2, we discuss some of the alternative operators tested. In addition, experiments using an ad-stream representation that encoded a page

Table 7. Partial score breakdown of 20 runs with 44-page initial ad distribution.

Score	# of pages	# of ads out of section	No page overflow
50559	43	0	√
50762	43	0	√
54773	43	0	√
57312	43	0	√
58094	43	0	√
58561	43	0	√
60064	43	0	√
61501	43	0	√
61964	43	0	√
62360	43	0	√
62621	43	0	√
62746	43	0	√
63147	44	0	√
64734	43	0	√
433662	43	5	√
618071	43	0	
620367	43	0	
650154	43	0	
703442	43	0	
755443	43	0	

Table 8. Final ad distributions for the best solution with a 43-page initial ad distribution and for the actual NYNEX layout.

# of pages away from section start	# of ads (algorithm)	# of ads (NYNEX YP)
0	60	59
1	29	20
2	24	15
3	9	23
4	20	22
5	10	13
Avg:	1.54	1.79

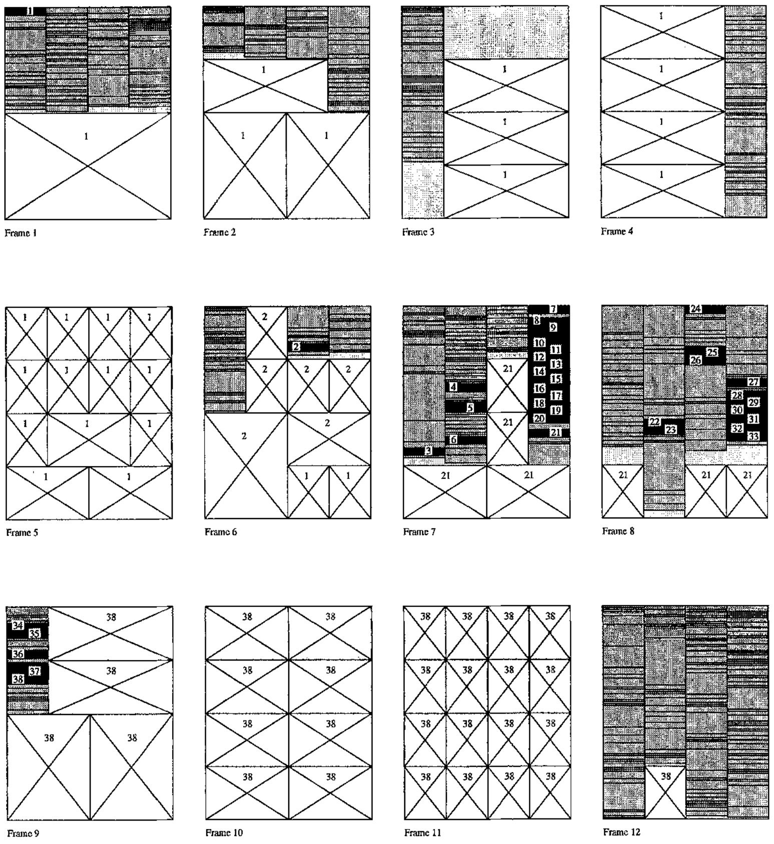
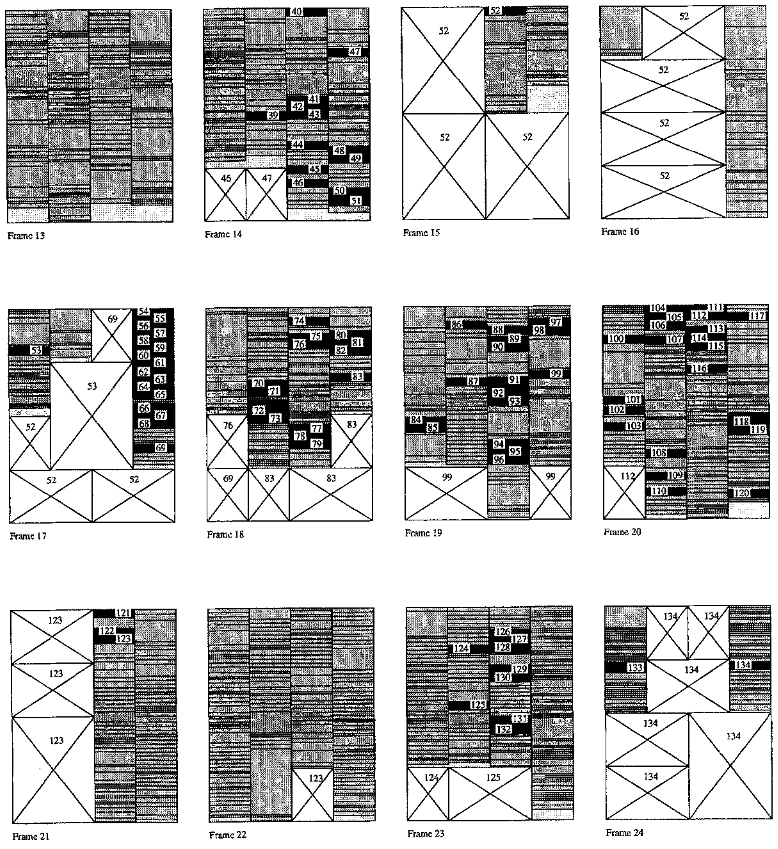


Figure 7.

directly for each ad rather than indirectly through ad-stream page breaks was also tried, with corresponding appropriate operators. Although this performed far better than the fully direct representation, it underperformed the ad-stream page-break representation described above.

We also addressed the efficacy of simulated annealing in relation to a simpler hill-climbing search strategy. By substituting the following parameter values into the pseudocode given in figure 3, one arrives at a simple stochastic hill-climbing search strategy: *TempFactor* = 1.0, *MinAccept* = 5, *StartTemp* = 0.0001, and *FrozenLimit* = 50. Thus unlike simulated annealing, hill climbing will never accept a perturbation that results in a worse solution. We performed 595 runs of hill climbing with a 43-page initial ad distribution, which took the same time as the 20 runs of simulated annealing reported in figure 5 and Table 6. Of these 595 runs, only four achieved scores below 100000. The best score was 65638, which is not in the top 60% of scores returned by SA.



Frame 13
Frame 14
Frame 15
Frame 16
Frame 17
Frame 18
Frame 19
Frame 20
Frame 21
Frame 22
Frame 23
Frame 24

Figure 8.

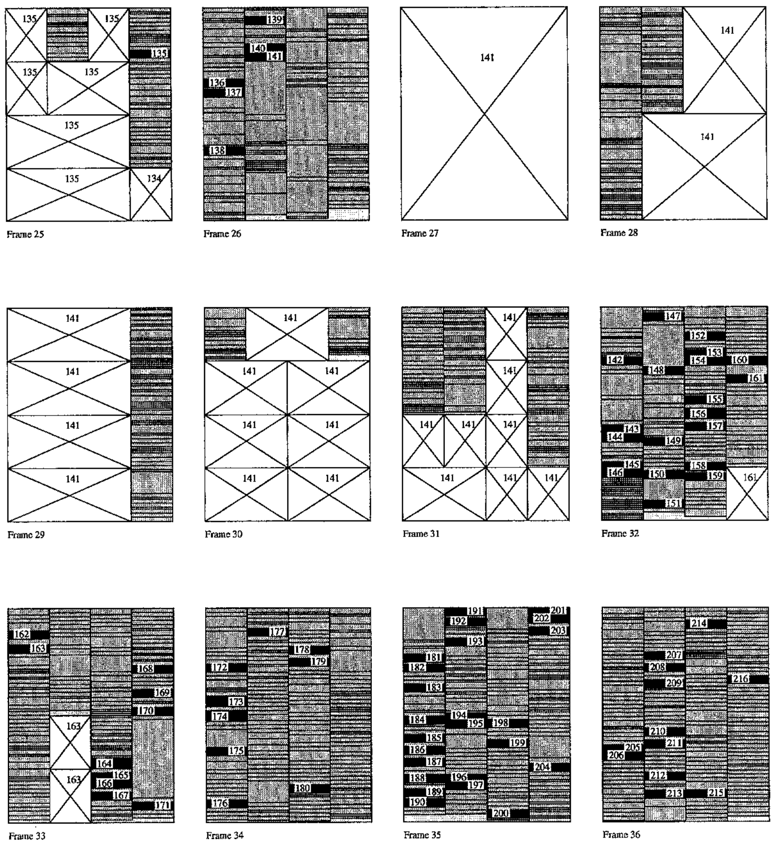


Figure 9.

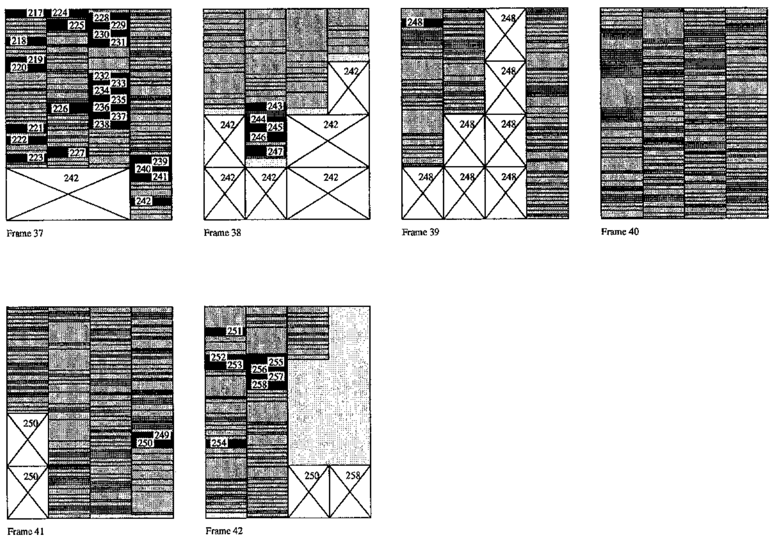


Figure 10.

5. Summary and conclusions

By utilizing an indirect representation for Yellow Pages layouts that can be decoded into full layout information, a simple stochastic search technique based on simulated annealing can generate extremely compact and harmonious page layouts, more compact, in fact, than those found in the actual published Yellow Pages by some 10% in our anecdotal experiments. Furthermore, since the method is based on a weak stochastic search technique, it is applicable to a broad range of page-layout tasks, as opposed to deterministic methods such as dynamic programming, which are both computationally intractable and too narrowly targeted. Nonetheless, care must be taken in building a stochastic search technique for Yellow-Pages layout; our experiments show the critical importance of problem representation in the effectiveness of the search.

Acknowledgments

We would like to thank Victor Milenkovic for helpful discussions, Karen Daniels for comments on an earlier draft, and Sandy Staff for assistance with library research. This research was supported in part by Grant Number IRI-9350192 from the National Science Foundation and by a grant from MERL—A Mitsubishi Electric Research Laboratory. We are grateful to U S WEST Advanced Technologies and Tony Cox on whose suggestion this research was undertaken.

Notes

Note in proof: It has come to our attention after this paper was typeset that a commercial YPPL software vendor, AMDOCS Inc. of St. Louis, Missouri, supplies software claimed to provide layouts of similar quality at speeds one to two orders of magnitude faster than those reported here. It is not known what methods are used by this software.

1. The idea of including a decoding step in the representation of candidate solutions comes from the evolutionary-computation community (Davis (1991)). This particular decoding process generates page layouts in which text is placed above ads within each column. This is consistent with standard practice, e.g., that of the Pacific Bell Yellow Pages. However, if desired, an alternative decoding process could be used, e.g., one that places ads atop each page, or one that attempts to interweave ads and text in a more complex way. The main requirements is that it achieve a compact arrangement of ads on a page.
2. The test data comes from pp. 266–312 of (NYNEX Information Resources (1995)), which spans roughly from "Carpet and Rug Dealers—New" to "Cleaning Service—Industrial" section, and comprises 3,817 entries and 152 ads. This data is available on the World-Wide Web at URL `http://www.eecs.harvard.edu/~shieber/YPPL/`.
3. The development data set came from pp. 1385–1435 of (NYNEX Information Resources (1994)), 46.25 pages that run roughly from "Restaurants" to "Rubbish & Garbage Removal", excluding the "Restaurants à la carte" section, which is a special four-page insert that is not in standard YP format. It comprises 3,450 entries and 159 ads, whose combined area takes up 39.3 pages. Allowing 2%–6% for filler therefore gives a range of 40–42 pages over which to distribute ads initially.

4. The values for the directory layout of the development data set were the same: 47 pages, no ads out of section, and no overflow. The computed results for the development data set are given in the following table:

Initial ad distribution	Number of satisfactory layouts (out of 20)	Breakdown of satisfactory layouts
40 pages	0 (=0%)	—
41 pages	13 (=65%)	41 pages (11) 42 pages (2)
42 pages	9 (=45%)	41 pages (6) 42 pages (3)

5. Many of the unsatisfactory layouts could be fixed by interleaving new pages and moving some of the contents of problematic pages onto them. An appropriate repair procedure could therefore increase the yield considerably.
6. The corresponding number for the development data set was 41 pages.
7. The corresponding numbers for the best result with the development data set were an average of 2.72 pages away from the section start for the computed layout, and an average of 4.01 pages away for the directory layout.

References

Černy, V. (1985). "A Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm," *Journal of Optimization Theory and Applications* 45, 41–51.

Chew, Hong-Gian, Moung Liang, Philip Koh, Daniel Ong, and Jen-Hoon Tan. (1994). "ALEXIS: An Intelligent Layout Tool for Publishing." In *Proceedings of the Sixth Annual Conference on Innovative Applications of Artificial Intelligence*. Seattle, WA, pp. 41–47.

Christensen, Jon, Joe Marks, and Stuart Shieber. (1995). "An Empirical Study of Algorithms for Point Feature Label Placement," *ACM Transactions on Graphics* 14(3), 203–232.

Davis, Lawrence (1991). *Handbook of Genetic Algorithms*. New York, NY: Van Nostrand Reinhold.

Furuta, Richard, Jeffrey Schofield, and Alan Shaw. (1982). "Document Formatting Systems: Survey, Concepts, and Issues," *ACM Computing Surveys* 14(3), 417–472.

Graf, W.H., S. Neurohr, and R.G. Goebel. (1996). "YPPS—A Constraint-Based Tool for the Pagination of Yellow-Page Directories." Technical Report, German Research Center for Artificial Intelligence (DFKI) GmbH.

Harada, Mikako, Andrew Witkin, and David Baraff. (1995). "Interactive Physically-Based Manipulation of Discrete/Continuous Models." In *Proceedings of SIGGRAPH '95*. Los Angeles, CA, pp. 199–208.

Iwai, Isamu, Miwako Doi, Koji Yamaguchi, Mika Fukui, and Yoichi Takebayashi. (1989). "A Document Layout System using Automatic Document Architecture Extraction." In *Proceedings of CHI'89*. Austin, TX, pp. 369–374.

Kirkpatrick, S., C.D. Gelatt, Jr., and M.P. Vecchi. (1983). "Optimization by Simulated Annealing," *Science* 220, 671–680.

Knuth, Donald E. and Michael F. Plass. (1981). "Breaking Paragraphs into Lines," *Software-Practice and Experience* 11, 1119–1184.

NYNEX Information Resources. (1994). NYNEX Yellow Pages: Boston area, area code 617.

NYNEX Information Resources. (1995). NYNEX Yellow Pages: Boston area, area code 617.

Peels, Arno J.H.M., Norbert J.M. Janssen, and Wop Nawjin. (1985). "Document Architecture and Text Formatting," *ACM Transactions on Office Information Systems* 3(4), 347–369.

Plass, Michael Frederick. (1981). "Optimal Pagination Techniques for Automatic Typesetting Systems." Ph.D. thesis, Stanford University.

Rosenking, Jeffrey P., Howard J. Marmostein, Eva M. Baron-Vartian, and Robert W. Soccio. (1991). "A Generic System for Directory Pagination." In *Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert System Programs*, pp. 166–169.

Ruml, Wheeler, J. Thomas Ngo, Joe Marks, and Stuart M. Shieber. (1996). "Easily Searched Encodings for Number Partitioning," *Journal of Optimization Theory and Applications* 89(2), 251–291.

Weitzman, Louis and Kent Wittenburg. (1994). "Automatic Presentation of Multimedia Documents Using Relational Grammars." In *Proceedings of the Second Annual ACM Conference on Multimedia*, San Francisco, CA, pp. 443–451.

Yellow Pages Publishers Association. (1995). *Yellow Pages Industry Facts & Media Guide*, 1995–1996 Edition.

Zoraster, Steven. (1991). "Expert Systems and the Map Label Placement Problem," *Cartographica* 28(1), 1–9, Spring.