Paginating the generalized newspaper – a comparison of simulated annealing and a heuristic method

Krista Lagus[1,2], Ilkka Karanta[1] and Juha Ylä-Jääski[1]

[1] VTT Information Technology, FIN-02044 VTT, Finland
(e-mail ilkka.karanta@vtt.fi)
[2] Currently on leave at Helsinki University of Technology,
Neural Networks Research Unit, FIN-02150 Espoo, Finland

Abstract. We consider the problem of placing ready-made boxes (containing e.g. articles) on the pages of a generalized newspaper. Three methods – one utilizing heuristic search, and two utilizing simulated annealing – are presented and compared. The second simulated annealing implementation emerged as the best method. It uses only one simple but powerful state transformation. In our view its success is due to clever use of domain knowledge in the design of the model.

1 Introduction

One important phase of the newspaper production process that has hitherto rather persistently resisted automatization is pagination, or the placement of articles and other material on the pages. There are several reasons for this, the main one beyond aesthetic considerations being that the problem is a two-dimensional bin-packing problem, and as such, np-hard.

There are several ways to characterize a pagination task. A central consideration is whether we can choose a selection from a large corpus of material or must all the material be paginated, and whether we can choose the scope (e.g. pages) of pagination or is the scope given beforehand.

The size of the problem naturally contributes to the complexity of the task. In the general case, the output of the pagination process normally consists of the chosen material together with page number and x and y coordinates for each item. There are several pages and several columns on each page, and the y coordinate may be a real number. A limitation in the range of any of these will reduce the complexity.

In this paper, we consider only pagination to a single page from a large corpus of editorial material. This is a fixed-space, varying contents problem. Such a problem emerges e.g. in the pagination of so-called fax newspapers. These are typically single page publications filled with actual articles from a newspaper, and faxed abroad to tourist centers, embassies etc. The problem emerges also when some section of the newspaper spanning only one page is being paginated. Furthermore, pagination to a single page can be considered as a step in paginating a whole newspaper.

A third potential application may emerge when markup languages start supporting multiple-column page layouts; this kind of enhancement is likely to appear in distributed hypertext systems of the future, such as the World Wide Web (WWW). If one then wants to create a WWW page with varying contents that fits on the screen of the user, these techniques will be readily applicable.

We describe and compare three approaches to the problem. One is based on a heuristic algorithm, whereas the other two utilize simulated annealing.

2 Background

Pagination with fixed size and varying contents involves two interrelated decisions: selection of *article forms* to be paginated to the page, and the actual placement of the article forms to create a page layout. An article form is a box with fixed size, shape and contents; one article may have many alternative article forms.

Not much literature exists on pagination problems. Our research group has a relatively long tradition in solving various pagination problems for yellow pages [2], newspaper advertisements [9], and generalized newspapers [8]. [7] describes the system that formed the basis for this paper. The search for optimal page layout was handled there in the heuristic way described in section 3. This system was then modified to allow for a simulated annealing search method, among other enhancements, and two simulated annealing approaches were implemented [6].

Pagination of article material can be handled as a generalization of two-dimensional multiple-knapsack problem (fixed space, varying contents) or as a two-dimensional bin-packing problem (fixed contents, varying space). Some literature for handling the latter kind of problems exists [5], but we have not been able to find any literature concerning the fixed space, varying contents problem outside the publications of our own research group.

The varying size, fixed contents problem can be treated as a *layout problem* [3], for which a number of optimal algorithms has been developed. However, due to the np-completeness of the quadratic assignment problem, these cannot be used to solve large-scale problems such as problems with fifteen or more items [4].

Several types of suboptimal algorithms have been designed to solve the layout problem. Some are based on heuristic search [7], some on simulated annealing (cf. [4]) and some on genetic algorithms [5]. The first two approaches will be discussed in the following sections. For the third approach the main difficulty has been encountered in developing a representation of the bin-packing problem that is suitable for a GA and is not limited to cuttings representable by slicing trees. However, according to [5], solutions found by heuristic algorithms can be improved by the application of GA's.

The three methods described in this paper were implemented to take care of the automatic pagination task of V·I·P^E, a system for automatic pagination of fax papers[3]. The input for V·I·P^{E}consists of a database of available material

[3] The heuristic method and the second of the simulated annealing methods were ac-

categorized into topics and sub-topics, a set of page frame descriptions (size, the number of columns etc.), a topic category for each page (or sub-page), and for each pagination task, an order from the customer, which directs both the selection of material and the search for best visual layouts. Based on this information V·I·P^{E}paginates selections of the available material to each page, also taking care of database management. The paginated pages may then be modified further via a graphical user interface, if deemed necessary. V·I·P^{E}also produces PostScript descriptions of the pages.

3 A heuristic method for pagination

The main problem in paginating articles with arbitrary size and (rectangular) shape is the combinatorial explosion. This led to the idea of reducing the problem size by discretizing the layout space not only vertically (which is done for newspaper articles anyway as they are presented in columns) but also horizontally. Typically, the page is divided into 4-6 horizontal sections, which together with the vertical division to columns constitute a sparse grid. The contents of each article are first paginated into several different modularized article forms by a rough layout. These forms are then passed to pagination, which selects at most one of them to a page.

The actual search algorithm is a mixture of best-first and branch and bound techniques. The status of the page at any state is represented as a graph that describes the places of the material inserted so far. This description forms the state of the search at each instant. In each state, the algorithm proceeds by searching through the empty spaces on the layout. Each maximal rectangular area of empty space is treated as a *region*, one possible position where to place an article form. Figure 1 illustrates the concept of a region.

At each state, the transition from a father to a child in the state diagram takes the following steps:

1. select N most suitable regions for pagination, and
2. for each such region, find out all the possible combinations of article boxes from the available article shapes in such a way that their combined width does not exceed the width of the region. Select M "best" combinations out of these.

Next, a number of selected region-box-combinations will be expanded to create new nodes of the search graph by inserting the box combinations to the regions. Each node is expanded further until no more articles can be added to the page. Finally, the best page according to given criteria is selected from all the states.

The heuristics of the approach reside in the ordering of empty regions (phase 1) and article combinations (phase 2) into a best-first order. It is difficult to formulate good rules for these ordering tasks.

---

tually implemented in place, whereas the first simulated annealing method was implemented only experimentally.

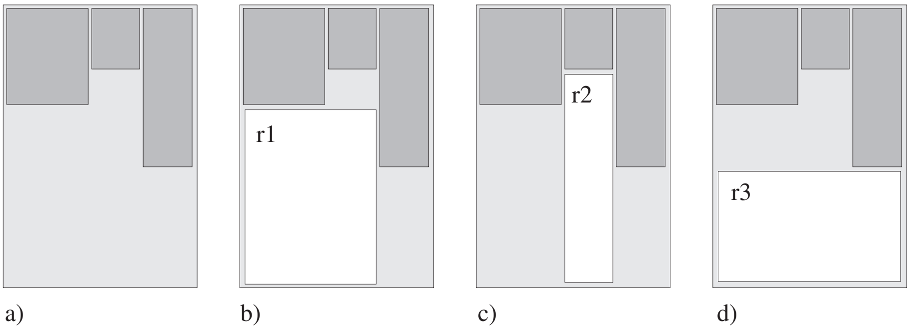

Fig. 1. a) A sample layout in the middle of pagination, with three articles placed. b), c) and d) The three maximal empty regions r1, r2 and r3 that can be found from the layout.

This approach has the merit that the resulting pages look tranquil. A major drawback is that the approach occasionally results in quite a lot of empty space on the page. This is due to two factors. First, because of the modularization, the article forms often contain empty space. Second, the actual search cannot be guaranteed to find a solution with minimal space usage given the article forms, because many alternative paths have been cut during the search based on the heuristic rules just described.

4 Application of simulated annealing

Simulated annealing is a stochastic method for selecting a state change, and within this context may be used to decide which change out of the possible ones to introduce to the layout at any time. First, a state is randomly selected from among the possible ones. If the state results in a lower cost, it is selected for expansion. If it results in a higher cost, it is with a certain probability chosen for expansion anyhow. This probability depends on a parameter called *temperature*, so that as the temperature is decreased the probability of selecting a worse solution becomes smaller. Finally, only changes improving the layout are allowed. The rate at which the temperature is decreased is called *cooling schedule*. For an excellent description of the method the reader is referred to [1].

In the following two subsections, we describe two methods for fixed-space, variable contents pagination that utilize simulated annealing.

4.1 Method SA1

In this approach, the two tasks of pagination – selection of material and creation of the layout – are separated, and the requirement that two distinct boxes do not overlap on the page is relaxed. First, an attempt is made to select material consisting of boxes with fixed size, shape and contents, and then annealing is used to find a configuration where the boxes overlap as little as possible. If the algorithm fails to find an acceptable solution, or one where the overlap is smaller than a predefined limit, the iteration is repeated with a new choice of material.

The selection of a suitable combination of article forms was implemented by utilizing a metaphor of an adder for numbers with variable base. Within this metaphor, each combination of digits (for example, 5366) represented one combination of article versions (version 5 of article 1, version 3 of article 2, version 6 of articles 3 and 4). The reason for variable base is evident: articles have varying numbers of versions.

The criterion chosen for ordering the trials of article combinations is to try to maximize the sum of article priorities (the article priorities were set beforehand manually for each article, and reflect its judged relative importance). This is accomplished by assigning the articles to digit positions in the order of their priority, and arranging the different versions of each article in the order of size from large to small. This ordering tends to maximize the amount of higher priority material on the page before descending to material of lower priorities, and produces an order roughly from best to worst in terms of total priority of the material on the page. Here we define total priority as

$$\sum_i \mathrm{area}(i) \times \mathrm{priority}(i), \tag{1}$$

where i runs through the prospective article forms.

Four criteria are used to prune out useless article form combinations before actually trying to place them on the page. The first criterion is that all the articles should, at least in principle, fit to the page judged by their area:

$$\sum_i \mathrm{area}(i) \leq \mathrm{area}(page), \tag{2}$$

where i runs again through article forms. Another criterion states that if the combined width of two article forms is greater than the width of the page, their combined height should be less than the height of the page. The other criteria are heuristic, and concern the minimum allowed area that is occupied by the material on page, and the minimum acceptable priority value for the whole page, respectively.

These criteria and the specific order of trial selections of material make it possible to prune whole branches of the search tree whenever the root of the branch is found unsuitable. For example, when constraint 2 is violated in some node of the search tree, it is not necessary to search through article form combinations whose combined area is even higher.

The actual search of article combinations that fit to the page involves the relaxation that the article forms may overlap on the page in the intermediate stages of optimization. The final requirement of no overlappings is coded into the cost function to be minimized,

$$\min E = \sum_{i,j,i\neq j} \mathrm{area}(\mathrm{box}(i) \cap \mathrm{box}(j)). \tag{3}$$

Fig. 2. An insert article transition during the annealing for a 5-column page. The shaded boxes are articles, darker boxes representing articles with higher priority value. The thick black lines stand for individual gaps, i.e. empty space.

State transitions were defined so that an article might at one transition move 1-30 rows within the same column, or alternatively move one column to either direction on the same row, with equal probabilities.

Annealing schedule for the search of article combinations was chosen so that in the initial temperature almost all attempts (over 90 %) were accepted. The end temperature was chosen so that less than 10 % were still accepted.

4.2 Method SA2

In this approach, the tasks of selection of material and finding the layout are integrated into a single optimization task. The approach utilizes a single transition insert article and replace if overlap occurs (called insert article in the following) that randomly chooses an article to be placed on the page, and also randomly finds a corner of empty space for it. Then articles that would overlap with this one are removed from the page, and the article is placed in the chosen empty spot.

In practice, empty spaces on the page are represented by regions described in section 3. An advantage of this representation is that regions give both form and size information that can be used in deciding where to put the chosen article. In addition, finding the regions in a given state is a relatively simple task, and it does not contribute much to the computational burden.

The operation of insert article are illustrated in figure 2. Here the insertion of article 31 results in an overlap with article 22, which is therefore removed.

The transition insert article has several important advantages. First of all, empty space cost is now a perfectly valid measure of the goodness of a state change. With this transition the neighbouring states will extremely rarely have exactly the same amount of empty space on the layout. Second, as no pure deletions are made, the process shows a quicker tendency to proceed towards full layouts. Third, the transition is powerful enough to produce a very great variety of page configurations (intuitively it is able to produce any conceivable article configuration on a page, but we do not have a formal proof of this). Fourth, the state transition scheme is comparatively simple which potentially renders it analyzable by formal methods. Fifth, increases in cost that result from state transformations tend to be small, and therefore chances of getting stuck at local optima are relatively small even in low temperatures.

The cost function was chosen on the grounds that in this approach, both minimizing empty space on the page and maximizing the sum of priorities of the articles on the page are essential. The cost function is then

$$E = U(A - \sum_i \mathrm{area}(\mathrm{box}(i))) - V(\sum_i \mathrm{priority}(\mathrm{box}(i)) \times \mathrm{area}(\mathrm{box}(i))), \qquad (4)$$

where A is the total available space on the page, and $U, V \geq 0$ are weights that tell how much we stress each single criterion.

In the average about 1500 rounds (i.e., suggested transitions) were used in annealing, with a final period of 30 rounds at zero temperature to further ensure being in at least a local minimum.

5 Comparison of the approaches

The heuristic method presented in section 3 is based on going systematically through a number of relevant configurations. Combinatorial explosion is avoided by restricting the form of relevant page configurations rather dramatically. This results in the relative efficiency of the implementation, but wastes layout space.

In our experience, a common pitfall of heuristic methods in pagination is that it is not easy to formulate heuristics that at the same time are sufficiently general, valid, and result in moderate run times. This stresses the criticality of available human resources since it is necessary to rethink the heuristics for each class of pagination tasks (for example, in extending our fax system to multi-page newspapers).

The simulated annealing method SA1 (section 4.1) resulted in awkwardly long annealing times for a fax newspaper. This is due to the vastness of the state space. The problem is underlined by the fact that when the articles to be put on the page are not known in advance, the tedious simulated annealing round must be repeated for every combination of articles that are to be tried. Another serious drawback of the method is that the resulting page layouts are not in many cases good or even acceptable (see figure 3).

Method SA2 (section 4.2) resulted in both acceptable run times and good quality page layouts. For example, paginating a set of 130-150 article forms 10 of which actually ended up on the page took less than half a minute on a Sun Sparcstation 20/m71 workstation. Run times were generally an order of magnitude shorter than with SA1. The actual difference in run times might be even larger since SA1 was implemented in C and SA2 in Common Lisp, which is usually considered to produce slower code than C.

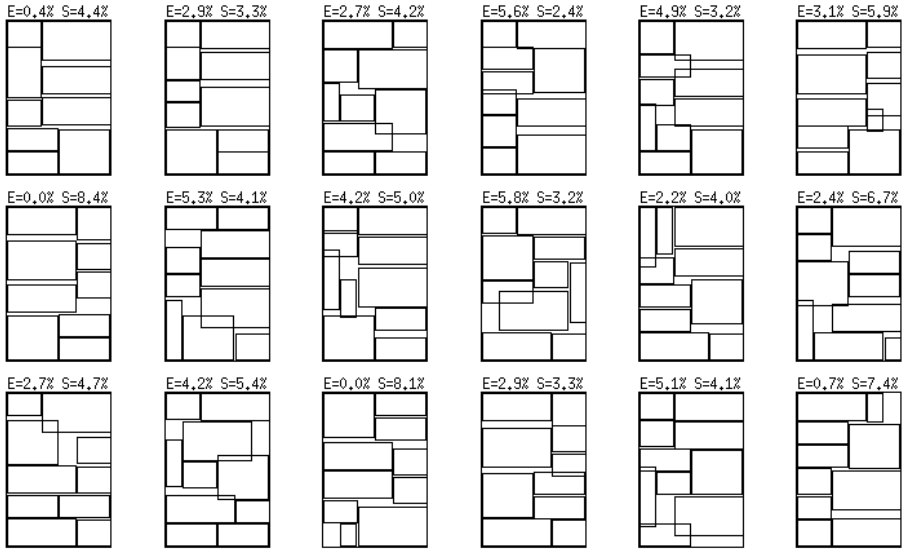

Fig. 3. Page layouts resulting from applying the method SA1. Notice the partial overlap of the articles.

The resulting page layouts were usually aesthetically pleasing. However, in some cases, the resulting pages were cluttered. The situation can be improved by post-pagination operations such as equalizing the spacings on the page automatically, but the fact remains that in some cases, manual reordering of the articles on the page or even repagination is necessary if aesthetics is a major concern.

A typical page layout produced by SA2 is presented in figure 4.

The good performance of SA2 is largely due to the design of state space and state transitions. Especially the transition insert article turned out to reduce the local discontinuity of the search space with respect to the cost function, which enabled considerable improvements even in the frozen annealing state.

Although SA2 is the most promising of the three approaches, it still is not quick enough for pagination that is a part of an interactive user interface. However, it can be used in the task it was designed for: pagination of a fax newspaper page that is delivered to the customers at most a few times a day.

6 Conclusions

We have presented and compared three methods, one based on heuristic search and two based on simulated annealing, in the application of automatic pagination of newspaper editorial material. The heuristic method is based on placing the boxes into a prechosen grid. The first simulated annealing method (SA1) is based on placing the boxes randomly and then applying simulated annealing to state

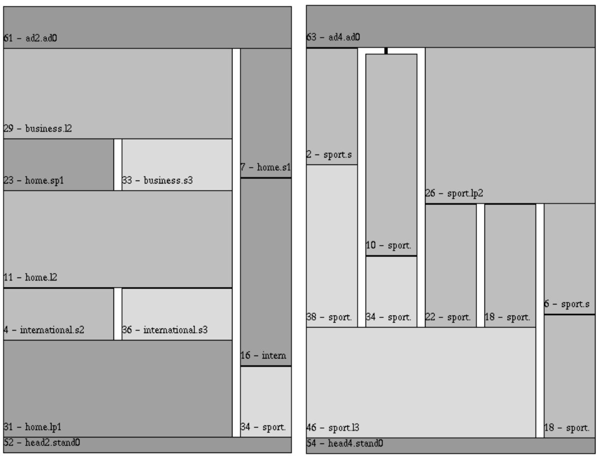

Fig. 4. Page layouts resulting from applying the method SA2.

transformations while trying to minimize overlap between articles. The second method (SA2) is based on applying simulated annealing with only one kind of state transformation: select a box to be placed on the page, find a suitable space for it, and remove any boxes that it would overlap on the page.

It is difficult to say whether better results in terms of speed and memory use could be obtained by using heuristic methods than when using simulated annealing. The main reason is that one can never say whether a better heuristic algorithm for the problem at hand could be designed if no currently known algorithm reaches optimal performance (and if one can define what the optimal performance is). Based on our long experience in development of heuristic methods for automatic pagination it seems evident, however, that using simulated annealing – or other search methods based on controlled randomization – will shorten development times of solution procedures considerably. This seems to be so since when using SA it is not necessary to investigate the problem area as thoroughly as when trying to end up with a good heuristic. Thus, SA provides an effective and well-analyzed vehicle of highly controllable randomization; the control is exercized when designing the state space and the cost function.

On the other hand, SA2 emerged as the winner of the two simulated annealing approaches. It seems to be no coincidence that SA2 is also the one that is based on a more thorough understanding and utilization of the special properties of the problem area. Therefore simulated annealing – or any other method – cannot be held as a substitute of genuine understanding of the problem. Rather, the degree of randomness brought by SA should be used in situations where it is difficult to set computationally simple and reliable rules to guide the search.

A future research direction is to combine the two approaches so that simulated annealing controls the global search, and heuristic methods guide local search in the neighborhood of an initial solution given by simulated annealing.

A demanding future research task is to find a means of analyzing the effects of state transitions. The analysis should both shed light on the performance and complexity of the methods and models in different pagination tasks, and on whether a particular search scheme is in principle capable of finding all possible page layouts from a given material.

Finally, two important research tasks concern the realm of subjective judgments. The first is to find an appropriate measure of the aesthetic appeal of page layouts. This could then be used in the cost function (or even in selecting which state expansion to take into consideration at each situation). The other is to find good ways to calculate the priority of an article, especially for a single user (or a group). This would allow to automatically customize the contents of the page for different customers. This topic is currently under research in our group.

References

1. E. Aarts and J. Korst. *Simulated annealing and Bolzmann machines.* John Wiley & Sons, 1989.
2. Hannu Ahonen and Petri Viertiö. Knowledge-based pagination. In Hannu Jaakkola and Seppo Linnainmaa, editors, *Proceedings of the second Scandinavian Conference on Artificial Intelligence*, pages 607–611, Amsterdam, Springfield Va., Tokyo, 1989. IOS Press.
3. S.S. Heragu. Recent models and techniques for solving the layout problem. *European Journal of Operations Research*, 57:136–144, 1992.
4. S.S. Heragu and A.S. Alfa. Experimental analysis of simulated annealing based algorithms for the layout problem. *European Journal of Operations Research*, 57:190–202, 1992.
5. S.-M. Hwang, C.Y. Kao, and J.-T. Horng. On solving rectangle bin packing problems using genetic algorithms. In *Proceedings of the 1994 IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1583–1590, Piscataway, NJ, USA, 1994. IEEE Press.
6. Krista Lagus. Automated pagination of the generalized newspaper using simulated annealing. Master's thesis, Helsinki University of Technology, 1995.
7. T. Veijonen, H. Ahonen, V. Leikola, J. Niku-Paavola, and J. Ylä-Jääski. Automatic page layout for electronic news distribution. In B. Furht, editor, *Proc. IASTED/ISMM International Conf. Distributed Multimedia systems and Applications*, Anaheim-Calgary-Zurich, 1994. IASTED/ISMM - Acta Press.
8. T. Veijonen, H. Ahonen, J. Niku-Paavola, and J. Ylä-Jääski. Knowledge-based system for the design of newspaper structure. In J. Liebowitz, editor, *Proceedings of the World Congress on Expert Systems*, pages 2172–2179. Pergamon Press, 1991.
9. J. Ylä-Jääski and H. Ahonen. Knowledge-based newspaper pagination. In *Applications of artificial intelligence VIII*, volume 1293, pages 767–774. SPIE, 1990.

This article was processed using the LaTeX macro package with LLNCS style