Layout Construction: A Case Study In Algorithm Engineering*

Gudmund Skovbjerg Frandsen
Jens Palsberg
Erik Meineche Schmidt
Steen Sjøgaard

August 1993

Abstract

We design a system for generating newspaper layout proposals. The input to the system consists of editorial information (text, pictures, etc) and style information (non-editorial information that specifies the aesthetic appearance of a layout). We consider the automation of layout construction to pose two main problems. One problem consists in optimizing the layout with respect to the constraints and preferences specified in the style information. Another problem consists in finding a representation of the style information that both supports its use in the combinatorial optimiza tion and supports its modification through high level user interaction and automatic inference from a database of examples.

We propose a solution that combines *heuristic search, randomization* and *neural networks*. We have implemented a first version based on the *bisection* strategy - a page is bisected recursively until the number of sub divisions matches the number of articles to be placed.

*This research was supported by CCI-Europe and the ESPRIT II BRA Programme of the EC under contract # 7141 (ALCOM 11).

1 Introduction

Page make up systems of current use in newspaper production has the basic function of replacing paper paste-up techniques. In addition these systems provide new tools such as continuous scaling of font sizes, and the systems automate simple tasks that were once left to the typographer, e.g. the task of hyphenation. However, major tasks such as the relative positioning of articles are decided by the typographer.

We expect that more tasks may be automated by future systems. We discuss one such task in this paper, namely the construction of layout proposals for a whole page. Such a construction is based on

1. *editorial input:* Includes the priority of articles and the size of headlines, text and accompanying pictures.

2. *style information:* Includes all information that relates to the aesthetic appearance of a page layout as opposed to the semantic contents of a page. Style information can be divided in two main categories, feasibility constraints and quality preferences.

   A constraint may forbid widows or it may require the text of an article to fill the assigned page area to within a 5 % allowance. If two distinct layouts both satisfy the constraints then style information of preferential type may rank one of them as superior in quality to the other. A few long text columns may be preferred to many short text columns and/or a page with a good balance in headlines and pictures may be preferred to a poorly balanced page.

The editorial input is easily included in the system, but the style in formation requires some care. We consider the automation of layout construction to pose two main problems:

1. *Optimization:* Given editorial input, the system must find a feasible layout that is optimal or near optimal with respect to a quality measure provided by the style information.

   From an abstract computational point of view, the editorial input is a vector of numbers $\mathbf{x} \in \mathbf{R}^n$, and so is a feasible layout proposal $\mathbf{y} \in$

$\mathbf{R}^m$. The style information restricts the feasible layouts to $L_x \subseteq \mathbf{R}^m$, and the style information also ranks the feasible layouts according to quality, $B : L_x \rightarrow \mathbf{R}$, i.e. the quality of a layout is measured on a linear scale as a real number. In general, we can not find a maximum quality layout due to combinatorial explosion. It is necessary to make a time/quality trade-off. The optimization problem consists in finding a high quality layout in L_x within a specified time limit. In the standard search terminology, our term quality would be mul tiplied by -1 and called *cost*. Hence, we would be looking for a low cost feasible solution [PaSt82].

2. *Handling of style information:* The chosen representation of style information must support its use as feasibility constraints and quality preferences for the optimization part. However, the representation must also support initialization and later modification of the style information either by explicit user specification and/or by more or less automatic inference of style information from a database of examples.

The paper is divided into two parts. In the first part, we present an abstract algorithmic solution to the two main problems. The solution uses ordinary algorithmic techniques in new combinations, but it uses no specialist layout knowledge. In the second part of the paper, we describe a concrete implementation of the abstract algorithm. The implementation is based on a specific strategy, bisection, for dividing a layout problem into sub-problems. We present various experimental results.

1.1 Abstract Algorithmic Solution

1. *Optimization:* We describe a kernel algorithm for the optimization. It is based on a standard heuristic state space search [HAI-1, **?**], which we augment with *randomization* and *time distribution*. The randomization is used as a way of getting nondeterministic behavior, i.e. the layout proposals generated by the system are independent of idiosyncrasies of the implementation, and multiple runs may result in several distinct layout proposals. The time distribution serves to make efficient use of a specified search time.

2. *Handling of style infomation:* The pruning and ordering of the search is based on feasibility and quality information. We represent that part of the style information that is used to specify quality preferences as a set of functions. The overall quality is defined as a linear combination of sub-functions, where each sub-function computes some value that is relevant for the quality. This is a flexible representation of quality information. The weights of a sub-function may be specified explicitly by the user or deduced implicitly by linear regression. Similarly, a sub-function can be specified by an algorithm, or it can be specified by a neural network that is trained to recognize a specific quality feature.

   The use of *neural networks* offers advantages in some cases. It can be a difficult and time consuming task for a programmer to understand and describe formally "holistic" aspects of quality such as the balance of headlines and pictures on a page. The programmer may be relieved from this task by letting a learning algorithm train a neural network to recognize the relevant quality aspects.

1.2 Implementation - the Bisection Strategy

We have implemented the kernel algorithm and made experiments with a specific strategy for dividing a layout problem into subproblems, *bisection*. When a number of articles are to be placed in some page area, we create two smaller problems by dividing the articles in two subsets and'we divide the page area in two by making a horizontal or vertical bisection. Clearly, this division of articles and page area can be done in many ways and the kernel algorithm controls the structured exploita tion of all possibilities. The restriction to axis parallel bisections may seem very crude, but it does comprise the layout used by a wide range of newspapers including *The Guardian* and *Financial Times*.

We have trained a neural network to recognize a specific "holistic" quality in layout proposals generated with the bisection strategy. The neural network is trained to prefer pages where small articles are placed contigously away from the center, and to dislike pages where small articles are randomly located throughout the page.

The experimental results are discussed in sections 3.4 - 3.5 and illustrated in Figures 1 - 13. In particular, Figures 4 - 6 show characteristic layouts generated by the system. The layouts in figures 10 - 13 were classified by the trained neural network.

2 A Kernel Algorithm

We have devised a template algorithm for layout construction. The template places a number of articles on the free part of a page by a divide and conquer strategy. At each step there will be many different possible divisions of the problem into subproblems involving fewer articles. The template has room for a *search space* module that specifies the divisions, which should be considered (line 1). A good search space module should satisfy the contradictory goals of making the search space small (only few possible next steps) and provide no unintended restrictions on the style.

There is also room for a number of *heuristic* modules that specify pruning (feasibility constraints, lines 2,5 - quality preferences, lines 4,5) and ordering (quality preferences, line 3) of the recursive search. The heuristic modules represent style information. The pruning and ordering of the search takes the greatest effect, when used early in the search. This leads to a classification of the style information, which we discuss later.

We have a special time distribution feature for controlling search within a specified time limit. In a subsection below, we explain how to plan ahead and spread the available time thinly out over the search space (the condition "$t \leq G_s(n)$" of the while-loop) rather than making a depth-first search that stops abruptly, when time is out.

In a separate subsection, we describe our use of randomization.

In a final subsection we discuss the advantages and disadvantages of using neural networks to represent style information.

First we present the kernel algorithm. The notation $(A \to f)$ denotes that the set of articles A is to be placed on the page part f. In an actual implementation of the kernel we use dynamic programming to avoid multiple calls of procedure *find layout* with identical arguments. For technical simplicity, we describe the kernel algorithm as outputting only one layout proposal (the best encountered). In practice, it would probably keep a number of the best proposals encountered and leave it for some kind of post-processing to make a final selection (see later).

procedure find_layout
input:
    A, a set of n articles.
    f, a free page part.
    (By assumption the combined areas of the articles in A
    equals the area of f.)
output:
    $(a_1 \rightarrow f_1, \ldots, a_n \rightarrow f_n)$, a high quality layout proposal.
    t, the number of time units used when computing the layout proposal.
method:
    $t := O$;
    so_far_best.quality $:= -\infty$;
    if $\mid A \mid = 1$ **then**
        so_far_best $:= (A \rightarrow f)$;
    else $(\mid A \mid >= 2)$
(1) $\mathcal{B}_1 :=$ The set of possible next steps $\subseteq$
        $\{(B \rightarrow g, C \rightarrow h) \mid$
            1. A is the disjoint union of the nonempty sets B and C.
            2. f may be divided in two parts g and h
        $\}$
        $t := t + \mid \mathcal{B}_1 \mid$;
(2) $\mathcal{B}_2 := \{b \in \mathcal{B}_1 \mid b$ is feasible $\}$;
(3) $\mathcal{B}_3 :=$ Priority queue containing the elements of $\mathcal{B}_2$
        ordered according to quality.
    while $t \leq G_s(n)$ and $\mathcal{B}_3 \neq \emptyset$ and
        the most promising next step $b = (B \rightarrow g, C \rightarrow h)$ from $\mathcal{B}_3$
(4)   can lead to an improvement of so_far_best
    do
        $(l_1, t_1 :=$ find_layout$(B \rightarrow g)$;
        $(l_2, t_2 :=$ find_layout$(C \rightarrow h)$;
        $l := l_1 l_2$;
        $t := t + t_1 + t_2$;
(5)   **if** l.feasible and l.quality $>$ so_far_best.quality **then**
            so_far_best $:= l$;

```
      fi;
    od;
fi;
return (so_far_best, t);
```

2.1 Randomization

If many distinct next steps from a single search state have identical quality ranks, then idiosyncrasies of the implementation will determine the order in which the corresponding problem divisions are investigated in detail before the time limit is reached. We use randomization to avoid this problem and to ensure a certain variation in the generated layout proposals.

The quality function is a linear combination of sub-functions. We let one sub-function return a random value independent of its argument. The user can control the amount of randomization by adjusting the weight of this sub-function. When the weight is nonzero but very small the quality ordering of layouts is preserved, and the randomization merely makes the output independent of the implementation specific search sequence. The system becomes in this sense nondeterministic. When the weight is moderately increased, the quality function becomes more random, and multiple runs on the same input produces a number of distinct layout proposals of the same flavor.

If we give the random function a relatively large weight in the total quality function, it may have the side effect that our greedy algorithm avoids a local optimum and obtain a better overall solution within the time limit. However, the randomization may just as well have the reverse effect and produce a worse solution. Simulated annealing offers a theo retical framework for using randomization when transferring an initially poor solution into a final good solution [San91]. A future search kernel may use simulated annealing.

2.2 Time Distribution

The time usage during the search is bounded by a search parameter s. To take a specific example, if $s = 3$, there will in every incarnation of the procedure be enough time for trying out at least three of the possible next steps recursively.

If a recursive call ends prematurely or too few next steps are available then the saved time is used to try more than three next steps in one or several ancestor incarnations.

The unit for our time measure is the time usage when handling a single possible next step including all lookahead computations (feasibility, quality). We assume that the number of immediate next steps $M(n)$ defined by the search space module depends on n only, where n is the number of articles to be placed. The total time consumed in one procedure call is the number of immediate next steps added to the combined time consumption of all recursive calls (We disregard the time used, when combining two partial layouts into a single as being negligible, similarly we ignore the time used in a call that places a single article, i.e. the case $n = 1$). If we pursue precisely s possible next steps recursively in every incarnation, we use at most time $T_s(n)$ to place n articles, where

$$T_s(n) = M(n) + s \cdot max_{i=1}^{n-1}(T_s(i) + T_s(n-i)) \text{for} n \geq 2$$

and

$$T_s(1) = O.$$

However, $T_s(n)$ is not the time limit that is checked at the beginning of each iteration (line 5). Before initiating another iteration we must be sure that there is enough time for the two recursive calls in the body of the while loop. Hence, we define

$$G_s(n) = T_s(n) - max_{i=1}^{n-1}(T_s(i) + T_s(n-i))$$

A calculation gives

$$T_s(n) = \sum_{i=0}^{n-2} s^i M(n-i)$$

and

$$G_s(n) = T_s(n) - T_s(n-1) = s^{n-2}M(2) + \sum_{i=0}^{n-3} s^i (M(n-i) - M(n-1-i))$$

for $n \geq 2$ and n integral, when assuming that $M(n)$ is increasing in n and $s \geq 1$ is a real number.

By stopping recursive calls when the time count exceeds $G_s(n)$, we try at least $\lfloor s \rfloor$ recursive calls (if that many next steps are available), yet we use at most time $T_s(n)$.

The user may control the time usage by specifying the search parameter s. The search time is expected to grow exponentially in s until most or all possible next steps are considered.

2.3 Partial layout vrs. complete layout

The search for a high quality layout runs through 4 phases:

top-down → single-article → bottom-up → full-page.

In the top-down phase a coarse layout is refined recursively. At the bottom of the recursion a single article is placed on the page. In the bottom-up phase several detailed layouts are combined into a single lay out. Finally we obtain the detailed layout for a whole page.

In all phases we discard possibilities and rank the remaining ones by means of style information. In the top-down phase the style information must relate to a possible next step problem division rather than to a detailed layout. Conversely, we may assume knowledge of all details in the bottom-up phase.

This implies that some layout concepts work well for the specification of style information that are used as search heuristics in one particular phase, while the same concepts work less well in other phases. However, heuristics take the greatest effect, when applied in an early phase:

When given a small search parameter s, the system has time to search only few full page layouts. Style information that relates to the top-down phase therefore dominates the output of the system. Conversely, if the search parameter is large, we expect to search through many detailed layouts, and so style information concerning the later phases has some influence on the output too.

2.4 Pruning vrs. Ordering in the Search

Style information can be used in three ways:

$$\text{static-feasibility-pruning} \\ \rightarrow \text{dynamic-quality-pruning} \rightarrow \text{quality-ordering.}$$

The most efficient search results from choosing a small set of im mediate next steps. We save lookahead computation by excluding an immediate next step from consideration, if this next step would later be rejected as non-feasible.

The static pruning of possible next steps based on feasibility information takes place before any of the next steps have been pursued recur sively. In comparison, the dynamic pruning based on quality information can only occur after the construction of a complete layout, since this type of pruning consists in ignoring those next steps that cannot possibly im prove an already found solution. Hence, static feasibility pruning takes effect earlier in the search than dynamic quality pruning, and feasibility information is thus more efficient than quality information.

If a partial layout of low rank is extended into a complete layout, it is likely to lose in the competition with other proposals. This waste of search efforts is avoided when using early pruning. Hence, it is more efficient to use quality information for pruning in addition to ordering than for ordering alone.

2.5 Using Neural Networks to Specify Style Information

The quality function is a linear combination of sub-functions that are represented independently. In this section we discuss the use of neural networks for computing selected sub-functions.

One of the major advantages of using neural networks consists of the more economical use of human resources. A programmer is relieved from the task of understanding and formally describing the specific function that must be computed, i.e. he does not have to concentrate on all the specific implementation details as he does when he is writing a traditional imperative program. Instead a learning algorithm "teaches" a neural net work to respond properly on a set of positive and negative examples which exemplifies the function or task considered. When training is completed, one hopefully possesses a neural network that generalizes well to exam ples outside the training set, i.e. the network responds properly to novel or unseen examples as well.

However, since a neural network "learns by examples," it selects and builds up by itself its own internal knowledge representation. In this sense a neural network can be considered as a black box, and this fact is of major importance to the applicability of neural networks. Suppose e.g. that a minor well-defined adjustment has to be done to a well-functioning neural network. The actual network may then be retrained on selected examples in order to add or build this new "behavior" into the existing network. However, due to the circumstance mentioned above, this extra "knowledge" will most likely affect the network's former knowledge about the problem domain, and thus unintended side effects of which one has only little control will be introduced. Therefore, application of neural networks should in general be restricted to atomic tasks, i.e. tasks for which there is no need to refer to subtasks and which do not have to be (re)adjusted.

On this background we expect that neural networks may be appro priate for representing "holistic" aspects of quality such as the balance of headlines and pictures. Given a set of course grained "pictures" of entire layouts, a neural network can be trained to identify those layouts that possess certain *overall* aesthetic qualities. In section 2.3 we point out that a quality function should be defined on partial layouts in order to influence the direction of search significantly. We could increase the usefulness of neural networks if it was possible to train a neüral network to make a good prophesy of the quality of a complete layout when input a partial layout only.

This more general application of neural networks is problematic. A neural network has a fixed number of inputs. In our case these inputs would probably correspond to a coarse grained image of a complete lay out. A partial layout would only define some of the inputs. We have to substitute dummy values for undefined values, and we must train the neural network on examples of both *partial* and *complete* layouts. But we can not expect to have any examples of partial layouts together with their prophesied quality values at our disposal. Q-learning offers a model for learning behavior from delayed rewards, which may be appropriate in this case [WaDa92].

3 Bisection, a Restricted Make Up Case

To illustrate the working of the conceptual framework, including the search kernel that we have described so far, we have chosen a sub-case of page make up. We use the bisection search strategy. Style information is specified by a number of simple layout concepts and a single "holistic" concept is defined by a neural network. The editorial input describes plain text and headlines. There will probably not be any new problems involved in extending the system to deal with more information of a se mantic nature such as the relative priority of articles. Similarly, balance problems connected with the placing of pictures have an analogue in the placing of headlines.

3.1 Search Space

The search space module allows only one kind of division, *bisection*. Bisection consists in dividing a free-area into two parts by a vertical or horizontal straight line. We consider all possible divisions of the set of articles into two subsets, each of which is assigned to one of the two sub-free-areas for recursive calls. Though bisection can not construct all possible layouts, it is surprisingly general. The bisection search space includes the layout styles used by *The Guardian* and *Financial Times*.

We let the search space module generate all possible bisections as immediate next steps. Hence $M(n) = 2(2^n - 2)$, and the time limit used in the kernel algorithm is

$$G_s(n) = \begin{cases} 2^n \cdot \frac{(\frac{s}{2})^{n-1}-1}{\frac{s}{2}-1} & \text{for } s \neq 2 \\ 2^n \cdot (n-1) & \text{for } s = 2 \end{cases}$$

This number grows exponentially fast in n. When dealing with a large number of articles ($n > 10$) it is necessary to provide additional heuristics to reduce the number of immediate next steps considered.

If several consecutive bisections are all vertical (or horizontal) then the same layout can be produced by permuting the order in which the consecutive parallel bisections are made. To avoid double work in this way we augment the search space module with "symmetry-elimination", which restricts the set of immediate next steps to enforce such layouts to be sought for only once.

3.2 Editorial Information

A page consists of a fixed number of columns.'

A *Free-area* is either the total part of a page, where articles are placed, or it is a component of a larger free-area. A free-area is wedge shaped, i.e. it takes the form of a rectangle, where zero, one or both lower corners has one or more sub-rectangles cut out (the cut out space may be reserved for commercial ads). We have chosen this class of free-areas because it is closed under bisection. The consideration of wedge shaped free areas only is not a severe restriction. In several international newspapers free-areas are in general wedge shaped.

Each article is characterized by its total area combined with the exact shape and size of the headline. The system places each article on a wedge shaped area that must allow room for the headline.

3.3 Style Information

We have based *feasibility* constraints and *quality* ranking on the following four layout concepts:

1. *area deviation:* The relative deviation of the actual area of an item in the layout compared to the area specified in the editorial input.
2. *text height:* The height of a text column (excluding headlines) in the layout. (minimum constraints on this parameter can be used to avoid

widows)

3. *article width:* The width of an article in the layout compared to the minimum possible, given that there must be room for the headline.

4. *direction variation:* The variation between horizontal and vertical bisections.

All the concepts may be used for both feasibility and quality specifications. In the test, we have carried out, we have used only two of the concepts for feasibility constraints. This is shown in Table 1 together with information about which of the four search phases may benefit from the use of heuristics connected to each concept. Quality ranking is used to order the search, but we have not implemented dynamic quality pruning.

If a feasibility/quality function is defined in the single-article phase, then its value in the bottom-up and full-page phases is computed by logical *and*, respectively arithmetic *sum*, from the values on a single article. Of course, the direction variation is not defined for a single article. Only area deviation and direction deviation are natural concepts to be used in the top-down phase.

concept	feasibility demands	quality prefers	top down	single article	bottom up	full page
area deviation	$\leq$max	small	X	(X)	(X)	(X)
text height	$\geq$min	large		X	(X)	(X)
article width	-	small		X	(X)	(X)
direction variation	-	large	X		(X)	(X)

Table 1: The applicability of the various layout concepts for expressing style information.

A layout is feasible, if it satisfies *all* feasibility demands. The user may influence the feasibility constraints by changing min/max values. The quality function is computed as a weighted sum of all quality sub functions. The user may adjust the weights. A weight of 0 ignores the sub-function in question and a negative weight reverses the preference large/small.

Our concepts supports an efficient implementation. They are all ap plied for heuristics in one of the first two phases (in addition to later phases). We can simplify the tables used in the dynamic programming part of the kernel algorithm considerably, because the feasibility/quality of a partial layout depends on the shape of the related free-area, but it does not depend on the exact position of this free-area.

On the other hand, we need concepts for expressing more holistic aspects of full page layout, and we would like to incorporate the use of neural nets into our prototypical system. We have addressed both needs by defining (informally) a quality function that prefers pages where small articles are placed contiguously at the sides of the page near bottom cor ners or near ads, and to dislike pages where small articles are randolllly located throughout the page. This function may be difficult to characterize formally and seems suitable for neural network computation.

3.4 Experimental Results with Kernel Algorithm

We have not integrated the neural network in the kernel algorithm. We describe the experimental results with neural networks separately at the end of this section.

We have made various detail optimizations on the kernel algorithm following the specialization to the bisection style. Dynamic programming is implemented for free-areas of rectangular shape only. The resulting system is quite efficient. Each sample layout (figures 4 - 6) was generated in less than one second, using search parameter $s = 3$.

Figures 1 - 3 show the editorial input symbolically, i.e. 7 articles with headlines and a page with a wedge shaped free-area. Figures 4 - 6 show different layout proposals generated by the system. We have defined three styles, a "flat" style that dislikes direction variation, a "variation" style that favors direction variation and requires text columns to have a minimum height and a "tall" style that favors long text columns and narrow articles. For each style we have chosen a characteristic layout from a set of 6 generated proposals. The examples illustrate how well one may control the style.

We have included a second example (figures 7 - 13) that illustrates the variation of proposed layouts within a specified style. Figures 10 - 11 and figures 12 - 13 show show two layouts of “flat” style and two layouts of “variation” style, respectively. In order to use the same example with the neural network experiments described below, the free-area (figure 7) has only 6 columns. This gives less freedom. However, by allowing a small deviation (5%) between the specified area for an article and the area actually assigned to the article we obtain more possible layouts, and combined with randomization, we obtain a satisfactory variation among the proposed layouts within each style.

3.5 Experimental Results with Neural Networks

The goal of this experiment was to train a neural net to prefer pages where small articles are placed contiguously at the sides of the page near bottom corners or near ads, and to dislike pages where small articles are randomly located throughout the page.

For technical simplicity we decided to focus on pages consisting of just 6 columns of text. However, an arbitrary part of the page could be occupied by commercial ads, and there was no practical restriction on the number of articles. The input to the net was a course grained image of the page (5 bits for each of the 5 pairs of adjacent columns) combined with a vector of real numbers that for each column specified the percent age of the area that was not occupied by commercial ads. This latter information ensured that the network had the necessary information to distinguish the ads from the articles. Each bit in the 5×5 binary image corresponded to a rectangular subfield of the page. The bit was set precisely when more than one article was visible in the corresponding subfield. This representation is a 2-dimensional version of the represen tation that Sejnowski and Rosenberg applied in the NETtalk experiment [SeRo86].

A large series of experiments were conducted which resulted in one hundred different networks. The training set consisted of 732 layout examples, while another 208 randomly generated layouts comprised the test set. All the networks were fully-connected two-layer feed-forward networks with short cut connections between the input units and the output units.

The generalization ability ranged from 80% for the poorest performing network to 94% for the best performing network, which had just 9 hidden units. This large variation in performance underlines the importance of trying out the same experiment several times. Due to different initial conditions (the initial weight values are selected randomly) the trained networks will only very rarely realize exactly the same function. However, in this case we were only interested in the best possible network, i.e. the one that was able to correctly classify 94% of the 208 layouts in the test set.

To illustrate the performance of the best network that we found, we presented the layouts in Figures 10 - 13 to the network. Recall that these were generated by the page make up algorithm in order to illustrate the usefulness of the randomization feature. The network correctly classified the layouts in Figures 11 and 12 as "good" (the small articles are placed at the bottom of the page in the outer columns or next to the commercial ads), while it disliked the layouts in Figures 10 and 13. In these experiments the neural network has been used as a post-processing module that accepts or discards the layout proposals generated by the search part of the system. However, it would be quite possible to inte grate the neural network as a heuristic module in the search algorithm and include the judgernent of the network with a weight like any other quality sub-function.

References

[HAI-1] *The Handbook of Artificial Intelligence, Vol 1*, editors: A. Barr and E. A. Feigenbaum, Pitman, 1981.

[PaSt82] Papadimitriou, C. H. and Steiglitz, K., *Algorithms and Complexity,* Prentice Hall, 1982.

[San91] Sangiovanni-Vincentelli, A., Editors Foreword (Special issue on Simulated Annealing). *Algorithmica 6* (1991) 295-301.

[SeRo86] Sejnowski, T. J. and Rosenberg, C. R., Parallel networks that learn to pronounce English text, in *Complex Systems I* (1987) 145-168.

[WaDa92] Watkins, C.J.C.H. and Dayan P.,
$\mathcal{Q}$-Learning.
Technical note. *Machine Learning 8* (1992) 279-292.

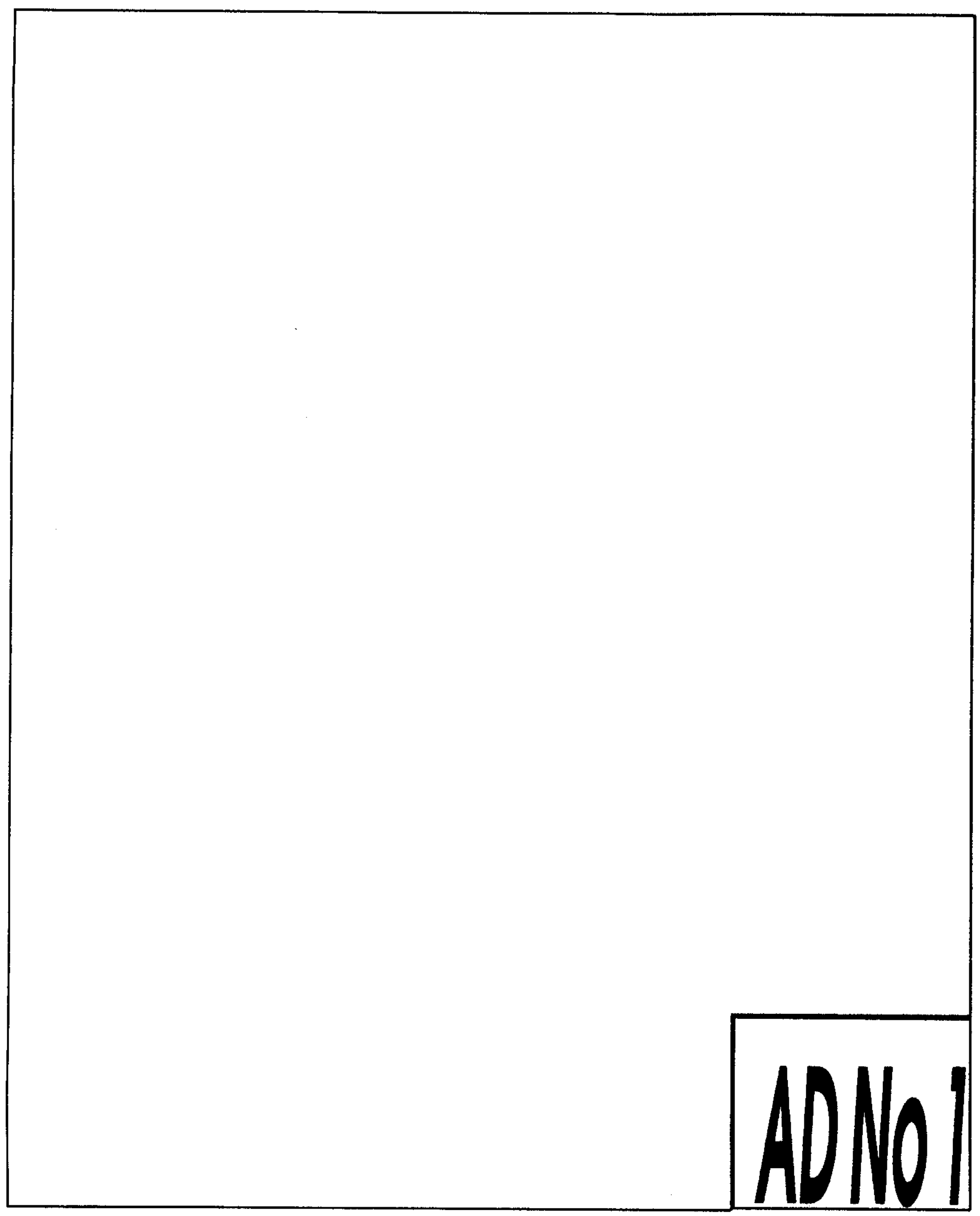


Figure 1: A wedge-shaped free-area.

Figure 2: Exact shape of headlines and total area for the articled 1-4.

ARTICLE No 5

Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking count-

ARTICLE No 6

Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to

ARTICLE No 7

Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the in-

Figure 3: Exact shape of headlines and total area for the articles 5-7.

Figure 4: A "flat" style layout.

Figure 5: A "variation" style layout.

Figure 6: A “tall” style layout.

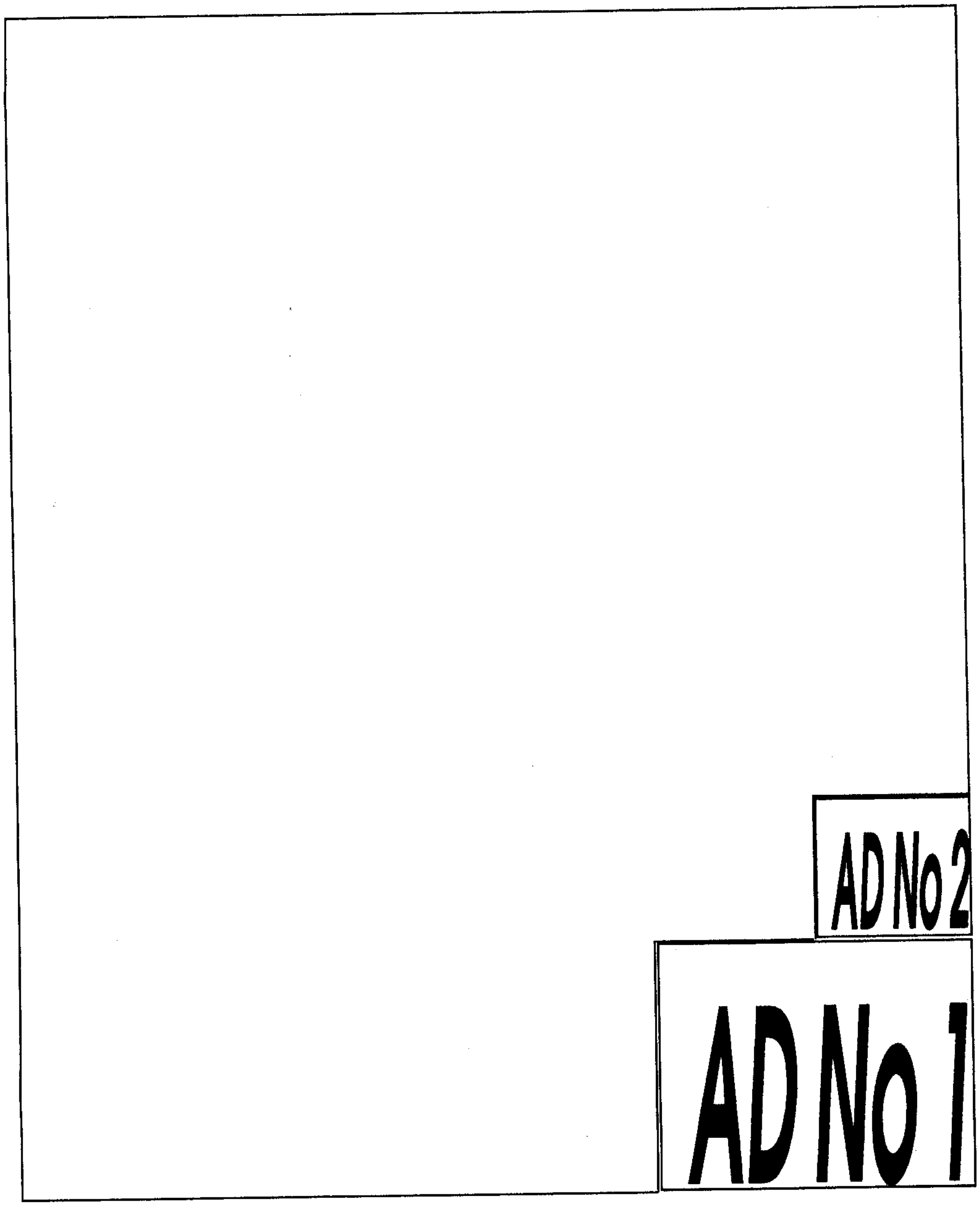


Figure 7: A wedge-shaped free-area (6 columns).

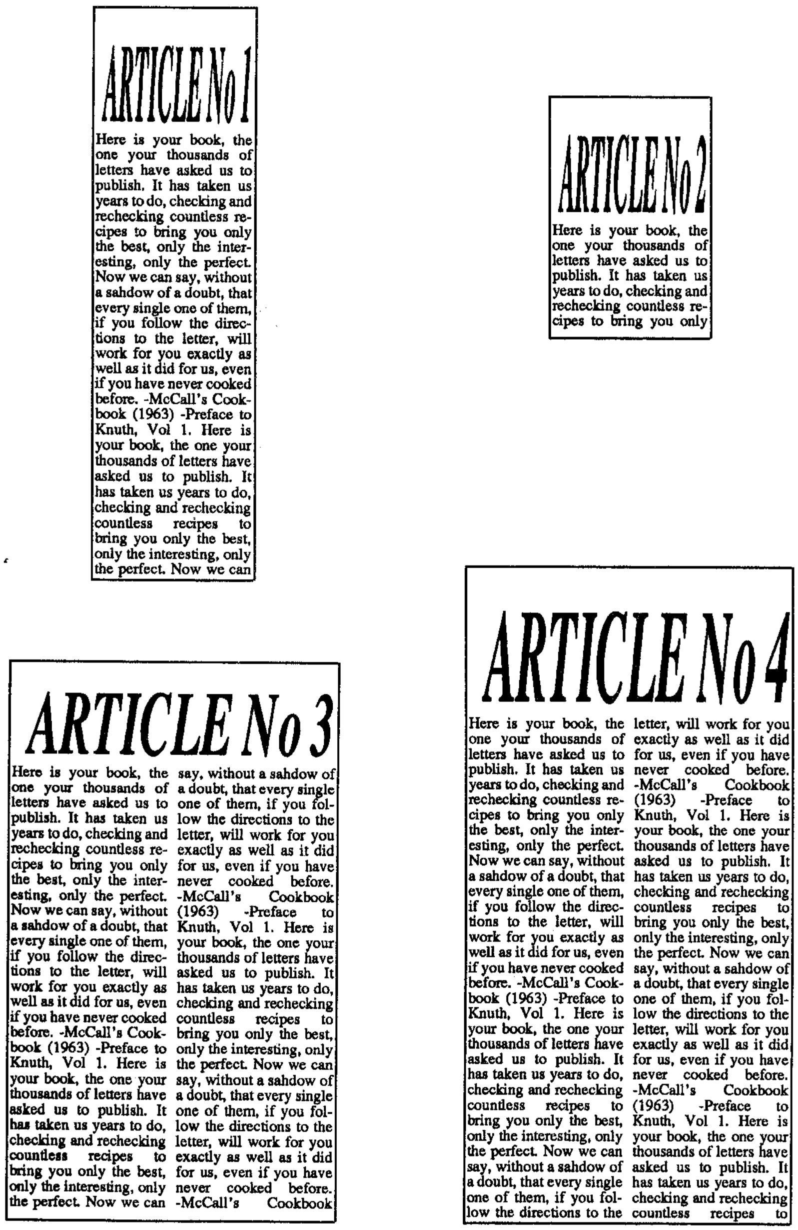
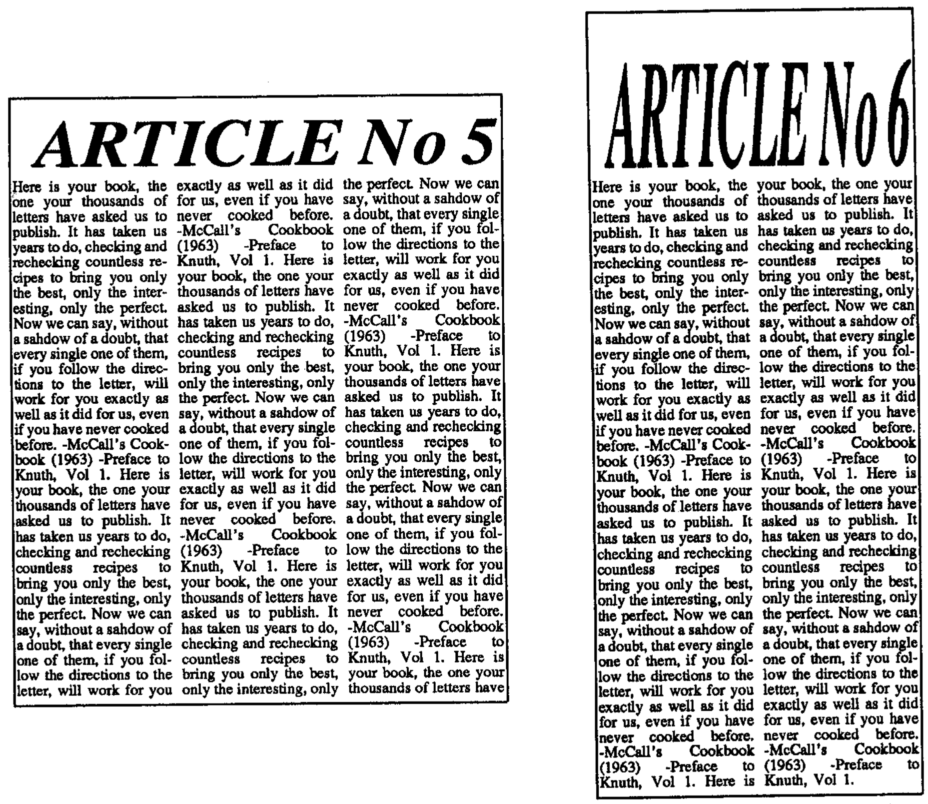


Figure 8: Exact shape of headlines and total area for the articles 1-4.



ARTICLE No 5

Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have

ARTICLE No 6

Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1. Here is your book, the one your thousands of letters have asked us to publish. It has taken us years to do, checking and rechecking countless recipes to bring you only the best, only the interesting, only the perfect. Now we can say, without a sahdow of a doubt, that every single one of them, if you follow the directions to the letter, will work for you exactly as well as it did for us, even if you have never cooked before. -McCall's Cookbook (1963) -Preface to Knuth, Vol 1.

Figure 9: Exact shape of headlines and total area for the articles 5 and 6.

Figure 10: A "flat" style layout. Judged as poor by the neural network.

Figure 11: A "flat" style layout. Judged as good by the neural network

Figure 12: A "variation" style layout. Judged as good by the neural network.

Figure 13: A "variation" style layout. Judged as poor by the neural network.