A heuristic solution of the rectangular cutting stock problem*

A. Albano and R. Orsini

Istituto di Scienze dell'Informazione, Università di Pisa, Corso Italia 40, 56100 Pisa, Italy

The rectangular cutting stock problem is to determine how to cut a specified number of each of certain types of rectangular pieces out of a given stock of rectangular sheets, the object being to minimise the waste. A special case of this general cutting problem encountered in many practical applications is one in which all cuts have to be accomplished from one edge of the rectangle to the opposite one, i.e. the cut has to be of a 'guillotine' type. For such applications the problem can be formulated as a mathematical optimisatiom program and the optimal solution can be found efficiently in terms of computer time only for problems of medium size. A heuristic method is described for an approximate solution of larger problems, which gives suboptimal results but presents an interesting tradeoff between computing effort and solution optimality.

(Received December 1978; revised March 1979)

1. Introduction

In many practical situations a specified number from a set of small rectangular pieces has to be cut from an unlimited number of larger rectangular sheets, the object being to minimise the waste, and thus the number of sheets used. Such problems appear in the cutting of steel, wood or glass sheets, and in other applications involving both cutting operations and situations unrelated to cutting such as the multiprocessing of a batch of programs. Depending on the technological process involved in cutting, and on the nature of the material to be cut, certain applications require the cut to be made along a straight line from one edge of the sheet to another. Such cuts, referred to in the paper as 'guillotine' cuts, are always required in the case of cutting glass, and quite often in the case of cutting wood or thin metals. In some circumstances the large rectangle to be cut contains defective areas and the cutting problem will be formulated to include restrictions on the area from which the pieces can be cut (Hahn, 1965); in others there are constraints on the sequencing of the cutting due to other operational criteria (Dyson and Gregory, 1974; Hinxman, 1977). This paper deals with guillotine cutting patterns, without constraints on the sequencing of the cutting, and for rectangular sheets without defects; the orientation of the pieces can be either fixed or left unspecified.

A related problem, the so-called template layout problem, in which no constraints are placed on the number of pieces of each type to be cut from a single sheet of material, has been studied by several authors. In some of the earliest investigations, Paull (1956), Eisemann (1957) and Vajda (1958) made use of linear programming techniques to solve a restricted version of the rectangular layout problem encountered in the cutting of rolls of paper. Haims and Freeman (1970) and Gilmore and Gomory (1966) give a dynamic programming algorithm of the template layout problem, and Herz (1972) used a tree search approach to avoid the exhaustive ones. These methods, however, cannot be generalised efficiently to deal with the constraints imposed on the number of pieces of each type to be cut. Another tree search approach was adapted to find the optimal solution of the cutting stock problem by Christofides and Whitlock (1977). Their procedure, however, is inefficient when solving problems of large size, as are often met in applications. In such cases, if the time to produce the solution is also of importance in evaluating the results, it is reasonable to consider a heuristic approach which does not yield the optimum layout but proves to be extremely efficient in terms of computer time. Of course a solution, in order to be valid, has to be on the average as near to the optimal one as possible. This approach has been used in the investigation reported here, and an algorithm is described which is an improved and extended version of the one proposed by Adamowicz and Albano (1976). The results shown indicate that the algorithm is a highly successful tool for the approximate solution of a wide class of cutting stock problems, that the produced solutions are quite close to the true optima, and that the computer demands are modest.

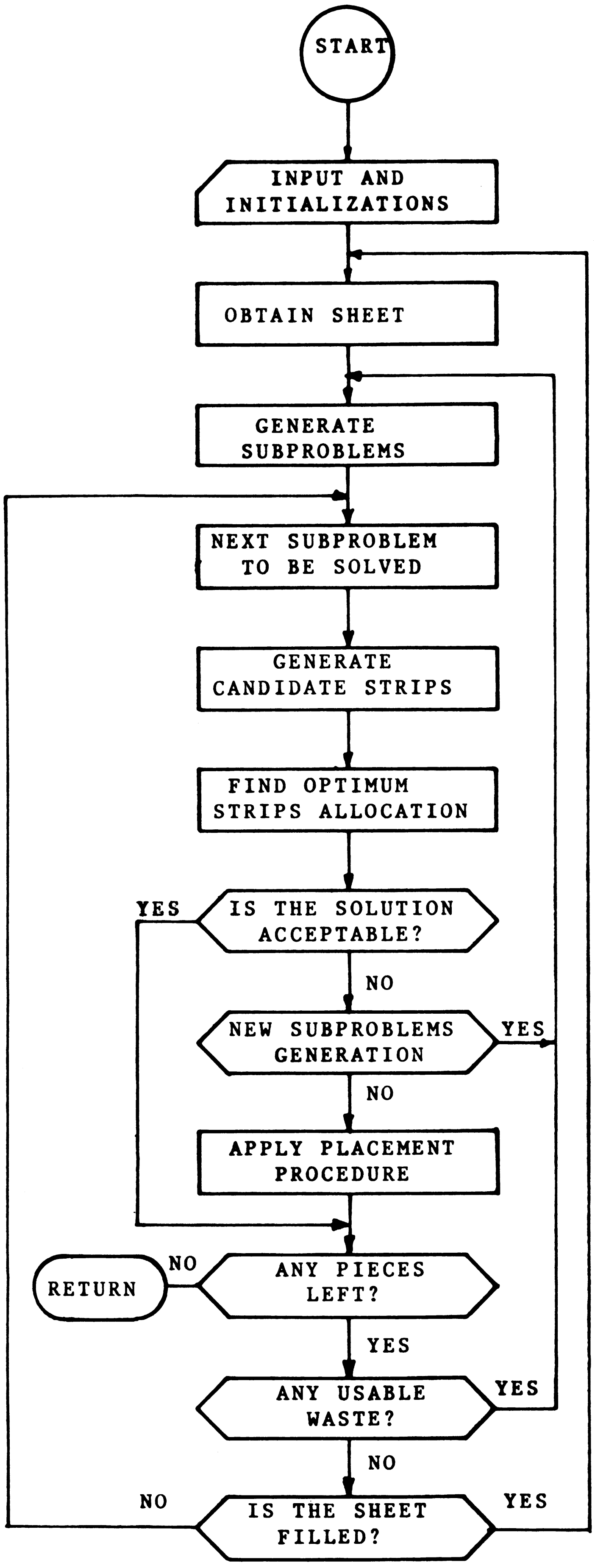
2. Solution approach

The approach we are going to describe is a combination of heuristic and exact techniques. The solution is found by a sequential decision making process based on certain heuristics that first decides whether to fill a sheet or to replace the original problem by two subproblems, obtained by a vertical guillotine cut, in such a way that their solutions will imply a 'good' solution of the original problem; then the current problem is approximately solved by allocating groupings of pieces called strips. If the solution is not satisfactory, either a different subproblem generation is decided or an approximate solution will be produced, applying a simple sequential placement procedure. During the process on the current sheet, at most three parts can be distinguished: the part that is already filled, the part currently under examination and the part still to be used. A new subdivision will not affect the first part, but it reduces the size of the second with respect to the third one. This process will continue as long as there are pieces to be allocated. The procedure is presented in **Fig. 1.**

In order to distinguish between the demanded pieces and the rectangular subpart of the sheet associated with the subproblem generated during the allocation process, the former are henceforth referred to as 'pieces' and the latter as 'rectangles'. The algorithm is based on the following main heuristics:

1. The sheets are filled one at a time. If they are of different size, they will be considered in the same order in which they are presented.
2. Once an acceptable layout of rectangles has been found for a sheet or for a rectangular subpart of it, this solution will not be reconsidered. In other words the remaining decisions do not influence a previously solved subproblem.
3. In order to generate a suboptimal arrangement of pieces on a rectangle, the pieces are, in general, considered in groups called *strips*. This heuristic is very important in reducing the dimensions of the problem. In fact, as we will see in what

*This work has been partially sponsored by the Consiglio Nazionale delle Ricerche.



START
INPUT AND INITIALIZATIONS
OBTAIN SHEET
GENERATE SUBPROBLEMS
NEXT SUBPROBLEM TO BE SOLVED
GENERATE CANDIDATE STRIPS
FIND OPTIMUM STRIPS ALLOCATION
YES
IS THE SOLUTION ACCEPTABLE?
NO
NEW SUBPROBLEMS GENERATION
YES
NO
APPLY PLACEMENT PROCEDURE
RETURN
NO
ANY PIECES LEFT?
YES
ANY USABLE WASTE?
YES
NO
NO
IS THE SHEET FILLED?
YES

Fig. 1 Overall approach

follows, at each stage the two-dimensional allocation problem is reduced to the allocation of strips in the width direction of the rectangle. Its use is supported by the following facts:

(*a*) it is adopted by an experienced man to solve this kind of problem by hand, and

(*b*) these kind of groupings are present in the optimal mathematical solution produced for medium size problems.

In this section we will describe how to generate and allocate strips, and then we will discuss the other heuristics used to generate subproblems when the produced layout is not acceptable.

2.1 *Strip generation and layout*

The algorithm makes use of three types of strips. A 'quasi-uniform' strip that is a set of pieces positioned as shown in **Fig.** 2(*a*), within a prefixed, horizontal range. A strip that is called 'uniform' (see Fig. 2(*b*)), if all pieces have the same width. Finally, a strip that is called 'homogeneous' (see Fig. 2(*c*)), if the pieces are of the same type. Let us assume that the rectangle associated with the current subproblem has a length L and a width W. Let the demanded set of unallocated pieces be

$$p = \{(l[1], w[1], n[1]) \ (l[2], w[2], n[2]) \dots \\ (l[N], w[N], n[N])\}$$

with $l[i] \geqslant w[i]$ for $i \leqslant N$ and $l[i] \geqslant l[i+1]$ for $i < N$ where $l[i]$, $w[i]$ and $n[i]$ are respectively the length, the width and the number of pieces of the same type to be allocated. Finally, let A be the area of the largest piece, Tn^*A a lower bound on the area of pieces to be used in creating strips and Tl^*L and Tw^*W two lower bounds on the strip length and width.

A set of candidate quasi-uniform strips is obtained with pieces from P by the following steps:

Step 1

Select the next piece $(l[i], w[i], n[i])$ in P with $n[i] > 0$ and compute the longest acceptable homogeneous strip length $L[i]$:

$$u1 = \max(l[i]^*m[i])$$
$$u2 = \max(w[i]^*m'[i])$$
$$L[i] = \max(u1, u2)$$

where $m[i]$ and $m'[i]$ are integers $\leqslant n[i]$, $l[i]^*m[i] \leqslant L$ and $w[i] \leqslant W$, $w[i]^*m'[i] \leqslant L$, $l[i] \leqslant W$ and $(l[i]^*w[i]) \geqslant Tn^*A$.

That is, if the piece can be rotated in the current rectangle, the preferred orientation is the one which produces the longest homogeneous strip.

Step 2

If $L[i] \geqslant Tl^*L$ and $w[i] \geqslant Tw^*W$ then go to Step 4. That is, only if the strip length and width achieve at least a certain specified percentage of L and W, the strip will be considered.

Step 3

Try to increase $L[i]$ up to the acceptable limit with pieces $(l[k], w[k], n[k])$ $1 \leqslant k \leqslant N$, $k \neq i$, and $w[k]$ greater than a specified percentage Th of the homogeneous strip width. If this is impossible, exclude the current strip and go to Step 1. Otherwise, continue with the next step.

Step 4

Include the current strip in the set of strip candidates to fill the rectangle. Compute the maximum number $b[i]$ of strips that

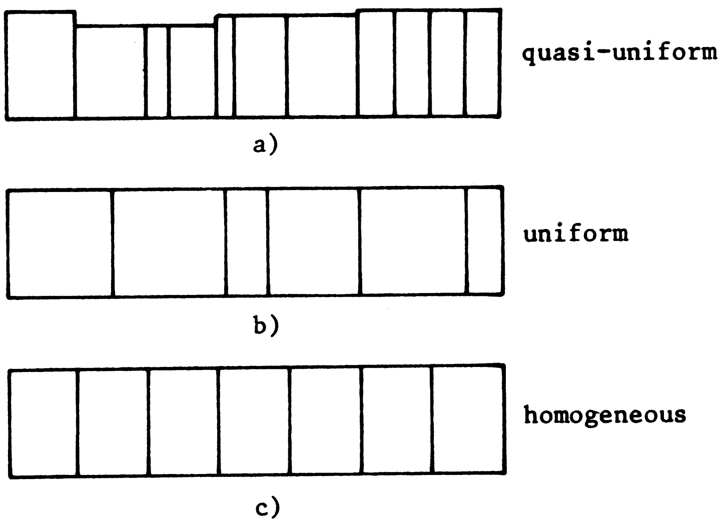


Fig. 2 Strip types

may be allocated without producing more rectangles than demanded, and the *strip value* $s[i]$ as the sum of the areas of the pieces used divided by $L[i]$. Then update the list P and go to Step 1.

Let $S = \{(w[i], b[i], s[i]), \ldots, (w[r], b[r], s[r])\}$ be the set of candidate strips. The values of the thresholds Tl, Tn, Tw and Th influence the number of candidate strips and therefore the capability to achieve acceptable layouts. Further comments about them will be given in the paragraph on the applications.

The problem of selecting a subset of S to maximise the total value of the strips allocated on the W wide sheet is the well known 'knapsack problem'. A classical method for its solution uses a dynamic programming technique to solve the following recursive function:

$$G(i + 1, x) = \max (G(i, x - k{*}w[i] + 1) + k{*}s[i] + 1) \quad i = 1, \ldots, r$$

where $0 \leqslant x \leqslant W, 0 \leqslant k \leqslant \min (b[i + 1], \lfloor W/w[i + 1]\rfloor)$*

The value $G(i, x)$ is the maximum value obtainable by allocating on a width x, strips from the first i of r allowable types. The function $G(i, x)$ is initialised so that $G(0, x) = 0$ is valid for all x. To compute $G(r, W)$, a table with W rows and a number of columns equal to $r + 3$ is required. In our implementation we have used a different approach based on a tree search algorithm with a memory requirement of $0(r)$, which is an extension of a proposal reported in Albano and Orsini (1980).

2.2 *Subproblem generation*

Let L and W be the dimensions of the rectangle associated to the current subproblem and let $v[i]$ be the *value* of a piece defined as the area $(l[i]{*}w[i])$, if it can be allocated both horizontally and vertically on the current rectangle, $v[i] = \max (l[i], w[i]){*}W$ otherwise. The strategies used to generate subproblems are as follows:

Strategy 1

Consider the piece $(l[i], w[i], n[i])$ with the *largest* value not yet allocated (from now on the 'reference piece') and form the longest homogeneous strip which can be allocated along the length direction of the rectangle. If the reference piece can be rotated, we have found it convenient to use the vertical orientation, i.e. the maximum extension in the 'width' direction. Let $L[k]$ be the length of the strips.

*The notation $\lfloor x \rfloor$ is used here for the value of the largest integer less than or equal to x.

If the rectangle $(L - L[k]){*}W$ is large enough to contain at least one of the remaining pieces, the original problem is reduced to the allocation of pieces on the rectangle $L[k]{*}W$ and $(L - L[k]){*}W$ (subproblem generation by length reduction). Otherwise this strategy is abandoned and the algorithm tries to allocate strips on the rectangle $(L{*}W)$ while the orientation of every piece will be decided during the strip generation. Finally, if either $L[k] = L$ or a subproblem generation took place, at least one strip formed by using the reference piece is forced to be included in the solution. The above definition of the piece value and this strategy have been introduced to allocate the pieces with greater surface as soon as possible.

Strategy 2

When the current subproblem has no acceptable solution, this is due to the fact that it has not been possible to generate enough strips to fill the width of the rectangle. Two types of techniques can be applied, depending on the number m of the *longest* piece (p), present in a strip.

Let

U	be the percent usage, i.e. the percent of the ratio between the total area of pieces allocated and the area of the current rectangle,
l	be the dimension of the longest piece along the strip direction,
w	be the other dimension,
n	be the number of p's still left for placement.

If $m > 1$, compute

$$m' = \lfloor(U{*}m/100) + 1\rfloor \text{ and let}$$

$$r = \begin{cases} m' & \text{if } m' < m \text{ or} \\ m - 1 & \text{otherwise} \end{cases}$$

and generate two subproblems by length reduction on the rectangles $(r{*}l, W)$ and $(L - r{*}l, W)$.

If $m = 1$, $l > w$ and $l \leqslant W$, then p can be rotated and two subproblems by length reduction will be generated on the rectangles according to the previous strategy.

Otherwise this strategy is abandoned and Strategy 5 is applied.

Strategy 3

During the process, the length reduction technique can occur several times on the initial rectangle. However, once a partial solution has been found and a new subproblem must be

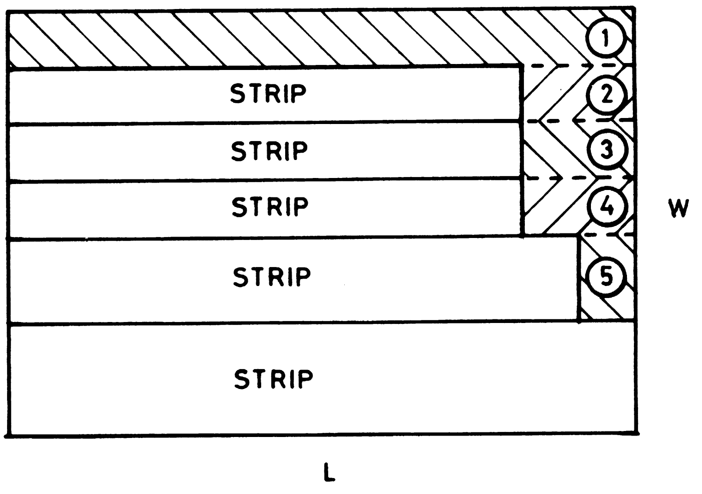


Fig. 3 Waste region examples

considered, the associated rectangle is obtained combining the previous adjacent rectangle with the same width to form a larger rectangle, which is more likely to be usable than the separate smaller one.

Strategy 4
Once a problem is solved some other subproblems may be generated, if the waste regions present in the current solution are large enough to be used to allocate some of the available pieces. The procedure for locating rectangular waste regions in the current solution is as follows. Assume $S[1], S[2], \ldots, S[k]$ to be the strips used in the solution and $L[i]$ and $W[i]$ to be the length and width of strip $S[i]$. The strips will be positioned packed together in order of their length with the longest strip on the bottom of the rectangle and with the leftmost edges of the strips flushing with the leftmost boundary of the rectangle. **Fig. 3** provides examples of waste regions. If two or more regions are adjacent and have the same length, they are combined to form a larger region. Regions 1, 2, 3 and 4 are examples of such regions. All the regions, which are potentially usable, produce new subproblems.

Strategy 5
If for the current rectangle no acceptable solution has been found and a new subproblem cannot be generated, Strategy 3 is applied and an approximate solution will be produced allocating one piece at a time according to a simple sequential placement procedure which preserves the 'guillotine cut' property. At each step, the procedure assumes as next piece to be allocated the one with the *largest* value. If the piece cannot be rotated ($l[i] \geqslant w[i]$ and $l[i] > W$), it is placed as many times as possible in the width and length directions to produce a step-front (**Fig. 4**). Otherwise the piece is rotated and placed as many times as possible in the width direction and this vertical strip is duplicated as far as there are pieces left and the width of the step front is less than the piece length. In both cases, after completion of this step, the leftmost-lower corner of the rectangle has been used. In order to preserve the guillotine cut property, the two rectangles 1 and 2 shown in Fig. 4 are considered and only on the first one the sequential placement procedure is applied again, while the second rectangle will be treated as a new subproblem.

3. Experimental results
The proposed algorithm has been implemented in FORTRAN IV on an IBM 370/168. The data used for the test problems were obtained both from hand made layouts produced for different applications, and generated randomly.

Table 1

Name	*Number demanded*	*Length [mm]*	*Width [mm]*
1	96	450	270
2	48	400	270
3	16	1140	170
4	8	2500	300
5	8	2500	139
6	40	300	270
7	32	1500	270
8	32	270	110
9	8	1993	165
10	16	710	675
11	16	500	500
12	32	800	500
13	30	300	74
14	8	1360	160
15	8	1760	785

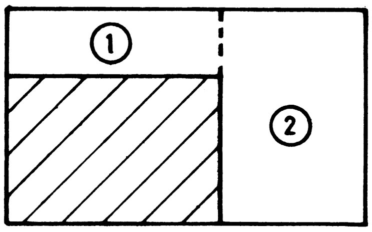


Fig. 4 Strategy 5: Subproblem generation

Table 2

Name	*Number demanded*	*Length [mm]*	*Width [mm]*
1	1	71	1708
2	1	71	403
3	1	71	439
4	1	69	174
5	3	66	174
6	10	69	167
7	30	66	167
8	1	69	706
9	3	66	120
10	10	438	120
11	50	438	119
12	1	4380	85
13	25	4380	70
14	1	69	353
15	14	63	352
16	1	846	73
17	1	846	70
18	1	1590	76
19	13	1590	64
20	3	69	908
21	1	192	71
22	9	192	62
23	30	64	66
24	1	87	112
25	11	86	112
26	11	87	107
27	121	86	107
28	1	68	1289
29	8	63	1289
30	10	1705	66
31	1	1705	85
32	1	1705	84
33	1	1705	111
34	2	79	1610
35	5	129	322
36	50	124	322
37	1	178	65
38	8	178	63
39	2	89	153
40	12	89	148
41	1	910	135
42	3	908	135
43	1	226	134
44	16	213	134
45	1	367	558
46	9	363	558
47	1	730	124
48	4	726	124
49	13	730	123
50	52	726	123
51	1	95	2550
52	1	81	2550

Tables 1 and **2** give the details of two sets of pieces to be allocated on standard types of sheet sizes of 12030 by 2550 mm. The first data set is the same as used in Adamowicz and Albano (1976) and has been selected in order to show the improved performance of the new algorithm. The second data set is an example of a class of tests that has been produced generating randomly a guillotine cutting pattern with no waste of the standard sheet as shown in **Fig. 5**.

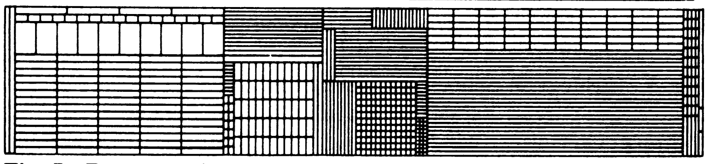


Fig. 5 Random guillotine cutting pattern with zero waste

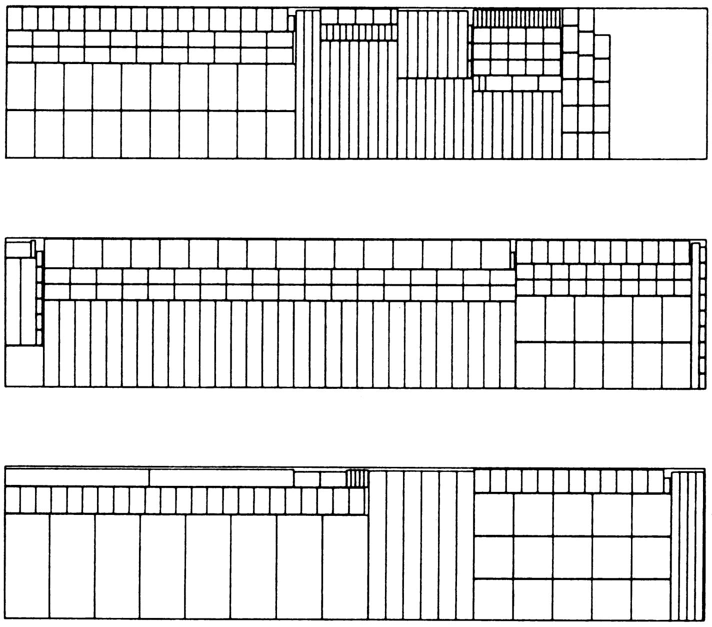


Fig. 6 Problem 1 layouts

The program requires, in addition to the piece and sheet descriptions, the values of the following parameters:

(*a*) The thresholds *Tn*, *Tl*, *Tw* and *Th* previously described, and the minimum percent usage *Tu* in order to obtain an acceptable solution.

(*b*) The option to use only homogeneous or quasi-uniform strips.

(*c*) The option to use the vertical or horizontal position for the reference piece when using Strategy 1.

(*d*) The option to force the reference piece in the current solution.

(*e*) The option to use the placement procedure provided by Strategy 5.

All parameters have standard default values and have been included to give the user the opportunity to adapt the algorithm to his data set. The influence of these parameters, both on the layout produced and on the execution time, has been experimentally investigated and the following choices have resulted, on the average, to give the best results: to use Strategy 5, to force the reference piece in the current solution, to adopt the vertical position for the reference piece, to use the quasi-uniform strips, and to assign the values *Tn* = 5-15%, *Tl* = 90%, *Tw* = 10%, *Th* = 90% and *Tu* = 90% for the thresholds. *Tn* is the most difficult one to choose, because its influence on the solution depends also on the kind of pieces present in the data set. With similar pieces it has no influence at all, with greater variations in the size of the pieces the positive effect of this threshold is to postpone the use of the smaller pieces and to anticipate the use of the larger ones. However it is important to note that the influence of these parameters decreases when the size of the problem increases. This behaviour confirms that this approach is more effective to solve practical problems of large size. This characteristic of the algorithm has been confirmed on the data set generated randomly with an optimal solution and

Table 3

Problem	*Sheet size*	*Sheet needed*	*Number of pieces*	*Time [sec]*	*Total % usage*
Standard set					
1	12030 2550	3	398	0·5	98·49
2	12030 2550	6	2 × 398	0·6	98·75
3	12030 2550	9	3 × 398	0·9	98·29
4	6015 2550	6	398	0·68	98·13
5	36090 2550	1	398	0·46	98·28
Random set					
6	12030 2550	2	563	0·93	96·56
7	12030 2550	4	3 × 563	1·55	98·15
8	12030 2550	7	6 × 563	1·85	98·52
9	12030	10	9 × 563	2·28	98·90

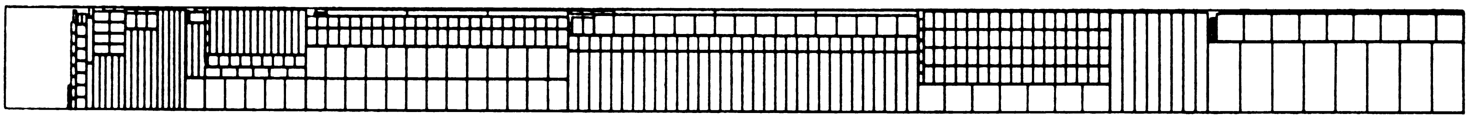


Fig. 7 Problem 5 layouts

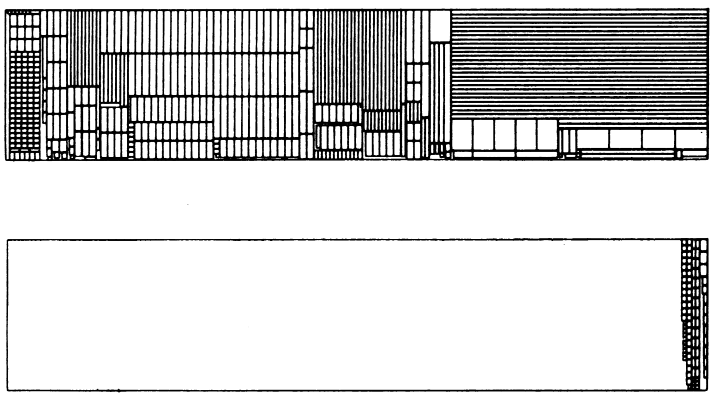


Fig. 8 Problem 6 layouts

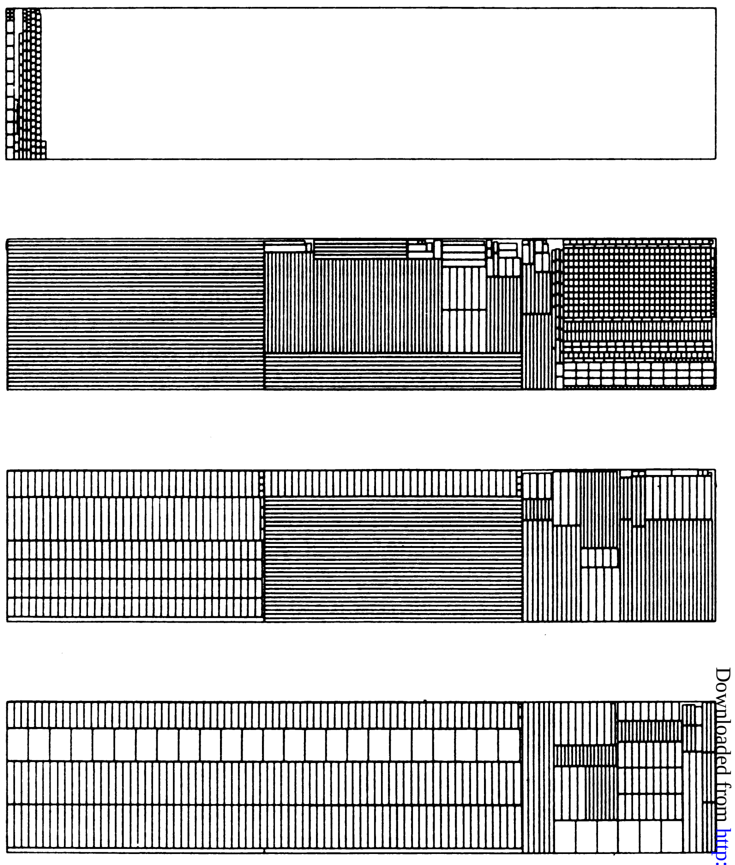


Fig. 9 Problem 7 layouts

no waste. The approximate solution presents a percent usage which increases when the number of pieces demanded is multiplied by a constant factor.

Table 3 summarises the results obtained when the algorithm is applied to some variations of the two data sets. **Figs 6, 7, 8,** and **9** show the cutting pattern produced for problems numbers 1, 5, 6 and 7. The performances reported for the random set can reasonably be regarded as average, because the algorithm has shown a stable behaviour.

As can be seen from the results, the method can be used to solve practical problems of large size and proves to be extremely efficient in terms of computer time and to produce solutions close to the true optima.

4. Conclusion

A heuristic method has been described to find an approximate solution of the optimal two dimensional cutting stock problem. Computational experience indicates that the algorithm compares favourably with respect to a previous procedure designed to solve this problem. The algorithm improves its performance with respect to solution time and optimality, by using the quasi-uniform strips, the vertical position for the reference rectangle, and the sequential placement procedure at the end of the allocation process. The results indicate that the algorithm is an effective tool for computing an approximate solution close to the optimal one and extremely efficient in terms of computer time for a wide class of cutting stock problems.

References

Adamowicz, M. and Albano, A. (1976). A Solution of the Rectangular Cutting Stock Problem, *IEEE Trans. Syst. Man Cybern.*, Vol. 6, pp. 302-310.

Albano, A. and Orsini, R. (1980). A Tree Search Approach to the M-Partition and Knapsack Problem, *The Computer Journal*, Vol. 23, pp. 256-261.

Christofides, N. and Whitlock, C. (1977). An Algorithm for Two-Dimensional Cutting Problems, *Oper. Res.*, Vol. 14, pp. 30-44.

Dyson, R. G. and Gregory, A. S. (1974). The Cutting Stock problem in the Flat Glass Industry, *Opl. Res.*, Q. 25, pp. 41-53.

Eisemann, K. (1957). The Trim Problem, *Management Sci.*, Vol. 3, pp. 279-284.

Gilmore, P. C. and Gomory, R. E. (1966). The Theory and Computation of Knapsack Functions, *Oper. Res.*, Vol. 14, pp. 1045-1074.

Hahn, S. G. (1965). On the Optimal Cutting of Defective Sheets, *Oper. Res.*, Vol. 13, pp. 94-120.

Haims, M. J. and Freeman, H. (1970). A Multistage Solution of the Template-Layout Problem, *IEEE Trans. Syst. Sci. Cybern.*, Vol. 6, pp. 145-151.

Herz, J. C. (1972). A Recursive Computing Procedure for Two-Dimensional Stock Cutting, *IBM J. Res. Develop.*, Vol. 16, pp. 462-469.

Hinxman, A. I. (1977). A Two-Dimensional Trim-Loss Problem with Sequencing Constraints, *5th IJCAI-77*, pp. 859-864.

Paull, A. E. (1956). Linear Programming: A Key to Optimum Newsprint Production, *Pulp Paper Mag. Can.*, Vol. 57.

Vajda, S. (1958). Trim Loss Reduction, in *Readings in Linear Programming*, New York, Wiley, pp. 78-84.

Visiting graduate teaching positions at the University of Helsinki

The University of Helsinki is to organise an intensified programme of graduate training and postgraduate research in computer science during the academic years 1981-1983. This programme is planned for implementation in co-operation with American and British computer scientists. Visiting scientists would stay in Helsinki for 4-8 months. It is also hoped that they will be able to advise their Finnish counterparts at the Computing Centre of the University of Helsinki.

Scientists interested in participation in the program are requested to contact Professor Martti Tienari, Department of Computer Science, University of Helsinki, Tukholmankatu 2, SF-00250 Helsinki 25, Finland.