An algorithm for the two-dimensional assortment problem

J.E. BEASLEY
Department of Management Science, Imperial College, London SW7 2BX, England

Abstract. In this paper we consider the two-dimensional assortment problem. This is the problem of choosing from a set of stock rectangles a subset which can be used for cutting into a number of smaller rectangular pieces. Constraints are imposed upon the number of such pieces which result from the cutting.

A heuristic algorithm for the guillotine cutting version of the problem is developed based on a greedy procedure for generating two-dimensional cutting patterns, a linear program for choosing the cutting patterns to use and an interchange procedure to decide the best subset of stock rectangles to cut.

Computational results are presented for a number of test problems which indicate that the algorithm developed produces good quality results both for assortment problems and for two-dimensional cutting problems.

Keywords: Heuristics, assortment, two-dimensional cutting

Received February 1983; revised February 1984

1. Introduction

The two-dimensional assortment problem is the problem that occurs when we are cutting a number of small rectangular pieces from large stock rectangles. Given the sizes of the various types of stock rectangles that we have available and details of the requirements for the small rectangular pieces the problem is one of deciding

(a) the appropriate types of stock rectangles to use (and how many of each type); and
(b) the two-dimensional cutting pattern for each stock rectangle that is cut into small rectangular pieces,

where there is a constraint upon the number of different types of stock rectangle that we can use.

This problem is encountered in industries concerned with the cutting of large (flat) rectangular items (typically the metal, glass and wood (e.g. furniture) industries) as they face the problem of choosing the best stock rectangles to cut in order to meet customer requirements.

In the next section we develop a mathematical formulation of the problem as a large integer program.

2. Problem formulation

In order to formulate the two-dimensional assortment problem we need to define a large number of factors relating to:

(1) the stock rectangles;
(2) the pieces to be cut; and
(3) the cutting patterns we can use.

We deal with these factors separately below.

(1) *Stock rectangles*

Let

n be the number of different types (sizes) of stock rectangles available,
L_i be the length of a stock rectangle of type i ($i = 1,\ldots,n$),
W_i be the width of a stock rectangle of type i ($i = 1,\ldots,n$),
f_i be the fixed cost associated with the use of a stock rectangle of type i for cutting ($i = 1,\ldots,n$),
k be the maximum number of different types of stock rectangle that we can use to produce the rectangular pieces, and
c_w be the cost per unit area of stock rectangle waste (i.e. each unit of area from a stock rectangle that is cut, but which is not part of some rectangular piece, incurs a cost c_w).

Informally we have n different types (sizes) of stock rectangles, costs associated with the use of

North-Holland
European Journal of Operational Research 19 (1985) 253–261

any stock rectangle and with any wasted stock rectangle area and a maximum number (k) of different types of stock rectangles that we can use.

(2) *Pieces*

Let

m be the number of different types (sizes) of rectangular pieces which we want cut from the stock rectangles,

l_j be the length of a rectangular piece of type j ($j = 1,\ldots,m$),

w_j be the width of a rectangular piece of type j ($j = 1,\ldots,m$),

v_j be the value associated with a rectangular piece of type j ($j = 1,\ldots,m$),

a_j be the lower limit on the number of rectangular pieces of type j that we cut ($a_j \geqslant 0$, $j = 1,\ldots,m$), and

b_j be the upper limit on the number of rectangular pieces of type j that we cut ($b_j \geqslant a_j \geqslant 0$, $j = 1,\ldots,m$).

Informally we have m different types (sizes) of rectangular pieces with a value associated with the production (cutting) of each rectangular piece. For the rectangular piece of type j the factor a_j represents the total demand for pieces of this type and $(b_j - a_j) \geqslant 0$ represents the number of pieces of this type that we are prepared to see cut for stock to meet future demand. Note here that enforcing $b_j = a_j$ ($j = 1,\ldots,m$) may result in wasted stock rectangle area which could be better used in producing pieces for stock.

(3) *Cutting patterns*

Let

$P(i)$ be the set of all two-dimensional cutting patterns for a stock rectangle of type i ($i = 1,\ldots,n$), and

d_{jpi} be the number of pieces of type j ($j = 1,\ldots,m$) that occur in cutting pattern $p \in P(i)$ ($i = 1,\ldots,n$).

We can now formulate the two-dimensional assortment problem as an integer program.

Let

x_{pi} be the number of times pattern $p \in P(i)$ is used for cutting a stock rectangle of type i ($i = 1,\ldots,n$), and

u_j be the number of rectangular pieces of type j ($j = 1,\ldots,m$) cut to meet the demand for pieces of this type.

Define

$y_i = 1$ if any stock rectangles of type i are used ($i = 1,\ldots,n$),

$= 0$ otherwise.

Then the program is:

$$\min \quad \sum_{i=1}^{n} \sum_{p \in P(i)} (f_i + c_w L_i W_i) x_{pi} - \sum_{j=1}^{m} (v_j + c_w l_j w_j) u_j, \tag{1}$$

$$\text{s.t.} \quad a_j \leqslant u_j \leqslant b_j, \quad j = 1,\ldots,m, \tag{2}$$

$$u_j \leqslant \sum_{i=1}^{n} \sum_{p \in P(i)} d_{jpi} x_{pi}, \quad j = 1,\ldots,m, \tag{3}$$

$$\sum_{i=1}^{n} y_i \leqslant k, \tag{4}$$

$$\sum_{p \in P(i)} x_{pi} \leqslant M y_i, \quad i = 1,\ldots,n, \tag{5}$$

$$y_i \in (0, 1), \quad i = 1,\ldots,n, \tag{6}$$

$$x_{pi} \geqslant 0, \quad \text{integer } i = 1,\ldots,n, \quad \forall p \in P(i), \tag{7}$$

$$u_j \geqslant 0, \quad \text{integer } j = 1,\ldots,m. \tag{8}$$

Equation (1) is the objective function, the first term in that equation representing the fixed cost of the stock rectangles used together with their associated area cost whilst the second term reduces this cost according to the value of the rectangular pieces cut and their corresponding area.

Equation (2) ensures that the demand for any piece is met. In equation (3) the right-hand side is the total number of pieces of type j produced by the cutting patterns adopted and the left-hand side of (3) is the number of pieces of type j used to meet the demand for pieces of this type. If (3) is not satisfied with equality then this implies that the cutting patterns adopted are producing more ($> b_j$) pieces of type j than can be used and the formulation of the objective function (1) ensures that such spare pieces are counted as waste area.

Equation (4) ensures that at most k stock rectangle sizes are used. Note here that we will have this equation satisfied with equality in the optimal solution of the program (equations (1)–(8)) given above (since if not we can artificially set some y_i to one to ensure that (4) is satisfied with equality without affecting the optimal solution).

In equation (5) M is a large positive constant and the equation ensures that no cutting patterns

for a particular stock rectangle are used unless the stock rectangle is one of the k sizes we are going to use. Equations (6), (7) and (8) are the integrality constraints. We have assumed here that there are no constraints upon the number of stock rectangles of each size available. Such constraints can, in fact, be easily incorporated into the formulation of the problem given above.

Note here that the objective function (1) can be simplified by defining

$$F_i = f_i + c_w L_i W_i, \quad i = 1,\ldots,n, \tag{9}$$

$$V_j = v_j + c_w l_j w_j, \quad j = 1,\ldots,m, \tag{10}$$

whereupon the objective function becomes

$$\sum_{i=1}^{n} \sum_{p \in P(i)} F_i x_{pi} - \sum_{j=1}^{m} V_j u_j. \tag{11}$$

It is well known that the number of possible two-dimensional cutting patterns $|P(i)|$ for any stock rectangle i can be very large and so the two-dimensional assortment problem as formulated above is a very large integer program. Hence it is clear that it would not be possible to optimally solve this program except for trivially small problems.

Note here that when the set of stock rectangles to be used are known (y_i known $i = 1,\ldots,n$) then the problem formulated above reduces to a two-dimensional cutting problem—the problem of deciding the appropriate cutting patterns to use, out of those available, to produce the small rectangular pieces at minimum total cost. It is common with problems of this kind to restrict the cuts that can be made to be guillotine cuts (a guillotine cut on a rectangle being a cut from one edge of the rectangle to the opposite edge which is parallel to the two remaining edges so that the cut divides (or guillotines) the rectangle into two).

A further simplification that is often made is to take the objective to be the minimisation of trim loss (wasted stock rectangle area). In terms of our objective (equation (1)) this is $c_w = 1$, $f_i = 0$, $i = 1,\ldots,n$, and $v_j = 0$, $j = 1,\ldots,m$. See [1] and [7] for a discussion of the various approaches to two-dimensional cutting problems.

The formulation of the two-dimensional assortment problem given above can be regarded as a "classical" formulation involving, as it does, explicit consideration of all possible two-dimensional cutting patterns for the stock rectangles under consideration. Recently Dyckhoff [4] has considered the one-dimensional cutting stock problem and shown how the classical formulation of that problem (also involving explicit consideration of all possible cutting patterns) can be improved upon by adopting a different approach to the problem. As noted in [4] his approach can be generalised (quite easily) to the two-dimensional (guillotine) cutting problem.

We will not (for reasons of space) give here the formulation of the two-dimensional (guillotine cutting) assortment problem as derived by his approach but simply note that it is also a large integer program which it would not be possible to solve optimally except for relatively small problems.

In the next section we consider previous work on the two-dimensional assortment problem.

3. Previous work

The general two-dimensional assortment problem has been considered by relatively few authors in the literature. Hinxman [7], in his survey, identified only two pieces of previous work.

Page [8] considered the problem of cutting steel bars where only one small rectangular piece could be cut from each stock rectangle and the problem was to decide the types of stock rectangles to use. A dynamic programming relaxation of the problem was used to provide a basis for a heuristic procedure to find the types of stock rectangles to use. A cost expression incorporating the purchase and delivery cost of stock rectangles, together with an inventory holding cost, was developed to decide the optimum number of stock rectangles of a particular type.

Chambers and Dyson [3] presented a heuristic procedure for a special case of the two-dimensional assortment problem. They used a procedure of Gilmore and Gomory [6] for calculating the minimum trim-loss to decide the width of all stock rectangles (all stock rectangles having the same width) assuming all possible stock rectangle lengths were available. Once the width was decided by this process the best set of lengths for k stock rectangles were picked from all significantly utilised lengths—an incremental procedure being used to slowly reduce the set of all significantly utilised lengths down to just k stock rectangle lengths.

They also presented a tree search procedure for selecting the best k stock rectangle lengths. Lower bounds to curtail the tree search were calculated from consideration of the available lengths at any tree node.

We note here that the one-dimensional assortment problem has been considered by a number of authors and significant savings in trim-loss have been found with appropriate stock sizes (see [5], [9]). We would expect that for the two-dimensional assortment problem significant savings could also be made by an appropriate choice of stock rectangle sizes.

In the next section we outline the algorithm we have developed for the two-dimensional assortment problem.

4. Algorithm outline

We noted before that as the integer programming formulation of the problem is very large (due to the large number of possible cutting patterns) we felt that we would not be able to optimally solve that program except for very small problems. Accordingly we adopted a heuristic approach to the problem.

As mentioned previously equation (4), limiting the number of different types of stock rectangles chosen, will be satisfied with equality in the optimal solution of the two-dimensional assortment program (equations (1)–(8)) i.e. exactly k different types (sizes) of stock rectangle will be used in the optimal solution. Hence we can regard the two-dimensional assortment problem as dividing fairly naturally into two separate problems:

(1) *Cutting pattern selection*

For a given set of k stock rectangle sizes decide the minimum cost cutting patterns to be used to produce the small rectangular pieces (l_j, w_j), $j = 1,\ldots,m$ (and hence the number used of each type of stock rectangle).

(2) *Stock rectangle selection*

Choose from all possible sets of k stock rectangles the best set of k stock rectangles (there being $n!/(n-k)!k!$ possible sets of stock rectangles).

We consider these two problems in turn and outline the solution approach adopted.

4.1. Cutting pattern selection

We decided to restrict attention to guillotine cutting patterns for two principal reasons:

(1) The vast majority of two-dimensional cutting problems met with in practise are guillotine cutting problems (e.g. see Abel et al. [1]). This is due to the nature of the cutting machines employed.

(2) Very little work has been done on generating non-guillotine cutting patterns (see [2]) but an efficient dynamic programming procedure (due to Gilmore and Gomory [6]) can be used to generate optimal unconstrained two-dimensional guillotine cutting patterns.

Hence for a given set of k stock rectangle sizes to decide the cutting patterns to be used we generated a number of different cutting patterns involving the m rectangular pieces which have to be cut and the k stock rectangle sizes we are considering. The linear programming (LP) relaxation of the two-dimensional assortment program ((1)–(8)) involving just (a) the k stock rectangle sizes we are considering; and (b) the cutting patterns generated above, was then solved with a heuristic rounding procedure being used to generate a feasible integer solution if the LP solution was fractional. At the end of this procedure we will have selected a set of cutting patterns which can be used to produce the small rectangular pieces (l_j, w_j), $j = 1,\ldots,m$.

The reasoning behind the above procedure is that by using LP to consider explicitly a (relatively) large number of (intelligently generated) possible cutting patterns we will be able to choose a good set of cutting patterns.

4.2. Stock rectangle selection

We considered that with $n!/(n-k)!k!$ possible sets of k stock rectangles it would not be practicable to examine all possible stock rectangle sets (except for k and n small). Accordingly we decided to select a subset of the n types of stock rectangles to examine in more detail. To do this we solved the cutting pattern selection problem, (as discussed above), with all n stock rectangle sizes available. The stock rectangles were then ranked in decreasing order of their utilisation and two sets K and S_T formed—where K is (essentially) the set of the k most utilised stock rectangles and S_T is (essentially) the set of the $(k+1)$, $(k+2),\ldots,(k+T)$ most utilised stock rectangles (some T). We then considered the interchange of stock rectangles from the set K with stock rectangles from the set S_T to see if an interchange led to an improved

solution. Such interchanges were evaluated using the heuristic for the cutting pattern selection problem discussed previously.

The algorithm terminates when no interchanges can be found which lead to an improved solution (or alternatively when a limit on computation time is reached).

The details of the algorithms for cutting pattern selection and stock rectangle selection are given in the next section.

5. Algorithm details

5.1. Cutting pattern selection

Let K be the set of stock rectangles available for cutting then the algorithm for cutting pattern selection is as follows:

(a) *Initial pattern generation*

For each piece j ($j = 1,\ldots,m$) and each stock rectangle i ($i \in K$) solve the unconstrained two-dimensional guillotine cutting problem consisting of:

(1) the stock rectangle i;

(2) piece j with value V_j (equation (10));

(3) piece q ($q \neq j$, $q = 1,\ldots,m$) with value V_q/M where M is a large positive constant.

This unconstrained two-dimensional guillotine cutting problem consists of finding a guillotine cutting pattern for the stock rectangle i that maximises the value of the pieces cut from it. Adjusting the value of all pieces q ($q \neq j$) as at step (3) above essentially means that we find the guillotine cutting pattern for i that contains as many pieces of type j as possible.

Note here that this unconstrained two-dimensional guillotine cutting problem can be solved by the dynamic programming algorithm of Gilmore and Gomory [6] and we shall not repeat the details of that algorithm here.

Computationally the calculation of the optimal cutting patterns for the k stock rectangles in K can be combined into a single calculation for the optimal cutting of a stock rectangle of length $\max(L_i \mid i \in K)$ and width $\max(W_i \mid i \in K)$. Note also that we need not repeat this initial pattern generation step for each set K of stock rectangles but need only carry it out once with $K = [1, 2,\ldots,n]$ and then simply recall any of the cutting patterns generated when necessary.

(b) *Greedy procedure*

Aside from the initial cutting patterns for any stock rectangle set K (generated as described above) we adopted a greedy procedure to generate some further cutting patterns. This procedure was as follows:

(1) Divide the m rectangular pieces to be cut from the stock rectangles in K into two sets Q_1 and Q_2 defined by

$$Q_1 = \left[j \mid a_j \geqslant 1, j = 1,\ldots,m \right], \tag{12}$$

$$Q_2 = \left[j \mid a_j = 0, j = 1,\ldots,m \right]. \tag{13}$$

Intuitively Q_1 contains pieces which must be cut and Q_2 contains pieces that need only be cut if they avoid creating stock rectangle waste.

(2) If $Q_1 = \emptyset$ then stop since we have no need to do any cutting else go to step (3).

(3) For each stock rectangle $i \in K$ solve the unconstrained two-dimensional guillotine cutting problem consisting of:

(i) the stock rectangle i;

(ii) piece j ($j \in Q_1$) with value V_j;

(iii) piece j ($j \in Q_2$) with value V_j/M.

As before these k unconstrained guillotine cutting problems can be combined into a single unconstrained guillotine cutting problem and can be solved by the dynamic programming algorithm of Gilmore and Gomory [6]. Intuitively we are generating guillotine cutting patterns that contain the pieces we need (Q_1 pieces) with the other pieces (Q_2 pieces) only being included in the cutting pattern if they avoid creating stock rectangle waste.

(4) Let A_i represent the value of the optimal unconstrained guillotine cutting pattern for stock rectangle $i \in K$ (as calculated above) and let e_{ij} be the number of times piece j ($j = 1,\ldots,m$) appears in that optimal cutting pattern.

(5) Choose the stock rectangle t where

$$F_t - A_t = \min\left(F_i - A_i \mid i \in K, e_{ij} \geqslant 1 \right.$$
$$\left. \text{for some } j \in Q_1 \right) \tag{14}$$

(ties broken arbitrarily). Intuitively we are choosing the stock rectangle t whose current cutting pattern of value A_t (containing at least one piece $j \in Q_1$ which we need to cut) most nearly covers the fixed cost F_t associated with the stock rectangle. We shall assume here that $F_t - A_t \geqslant 0$ (as will be the case in most practical situations) implying that the fixed cost of the rectangle outweighs the value of the pieces cut from it.

(6) We will use the optimal cutting pattern for stock rectangle t to exhaustion (a greedy procedure). The number of times s that we use this cutting pattern is given by

$$s = \min\left(\left\lfloor a_j/e_{tj}\right\rfloor \mid e_{tj} \geqslant 1, j \in Q_1\right) \quad (15)$$

where $\lfloor x \rfloor$ is the smallest integer greater than or equal to x. Intuitively we use the cutting pattern as little as possible consistent with exhausting the requirements for one of the pieces $j \in Q_1$.

(7) Update the problem by

$$a_j = a_j - \min(a_j, se_{tj}), \quad j \in Q_1. \quad (16)$$

Note that the $\min(a_j, se_{tj})$ term is needed in the event that se_{tj} exceeds the number (a_j) of pieces of type j that we need cut.

(8) Go to step (1) to resolve the problem. Note here that the definition of s (equation (15)) ensures that when we go to step (1) $|Q_1|$ will be reduced by at least one and hence that the entire procedure will terminate (step (2)) after at most m iterations.

(c) *Linear program*

As mentioned previously we solve the LP relaxation of the two-dimensional assortment problem ((1)–(8)) involving just

(i) the k stock rectangle sizes in K,

(ii) the cutting patterns associated with the stock rectangles in K as found by

(a) the initial pattern generation procedure; and

(b) the greedy procedure.

Formally let $P(i)$ be the set of all two-dimensional cutting patterns for a stock rectangle of type i ($i \in K$) as found by

(1) applying the initial pattern generation procedure for all pieces j ($j = 1,\ldots,m$) to the stock rectangle i; and

(2) step (3) of the greedy procedure (one cutting pattern for each iteration of the greedy procedure).

Note here that $|P(i)| \leqslant 2m$ when defined as at (1) and (2) above. Then the LP relaxation of the two-dimensional assortment problem (given the stock rectangle set K) is

$$\min \quad \sum_{i \in K} \sum_{p \in P(i)} F_i x_{pi} - \sum_{j=1}^{m} V_j u_j, \quad (17)$$

$$\text{s.t.} \quad a_j \leqslant u_j \leqslant b_j, \quad j = 1,\ldots,m, \quad (18)$$

$$u_j \leqslant \sum_{i \in K} \sum_{p \in P(i)} d_{jpi} x_{pi}, \quad j = 1,\ldots,m, \quad (19)$$

$$x_{pi} \geqslant 0, \quad \forall i \in K, \forall p \in P(i), \quad (20)$$

$$u_j \geqslant 0, \quad j = 1,\ldots,m. \quad (21)$$

This is a relatively small LP involving (at most) $3m$ constraints and $m(2k+1)$ variables and can be easily solved for quite large problems by a primal simplex procedure. Examination of early computational results showed that (almost always) the optimal solution of the above LP was non-integer (fractional). We adopted the following rounding scheme to cope with such cases.

(1) Let (X_{pi}) be the values of (x_{pi}) in the optimal solution of the LP.

(2) Round *up* every fractional X_{pi} value.

(3) At the end of step (2) it may be possible to reduce down certain X_{pi} values—thereby reducing the value of the objective function (equation (17)) whilst still meeting the requirements (equation (18)) for the m rectangular pieces. This can be done as follows:

(a) consider the X_{pi} in decreasing value order and for each $X_{rs} \geqslant 1$;

(b) decrease X_{rs} by as much as possible consistent with maintaining

$$\sum_{i \in K} \sum_{p \in P(i)} d_{jpi} X_{pi} \geqslant a_j \quad \text{and} \quad X_{rs} \geqslant 0.$$

(4) At the end of step (3) we will have a feasible integer solution to the two-dimensional assortment problem (given the stock set K).

Examination of computational results showed that often the value of the objective function (17) for the (rounded) integer solution was significantly higher than the value of the objective function at the LP optimum. Accordingly, we implemented the more sophisticated rounding scheme described below based upon cutting planes.

(d) *Cutting plane rounding*

As before solve the LP relaxation of the two-dimensional assortment problem. In order to move the LP solution towards an optimal integer solution we introduce a number of Gomory f-cuts (e.g. see [11]). This use of f-cuts was inspired by their use by Scarborough [9] in her work on the one-dimensional cutting problem. The procedure we adopted was as follows:

(1) Generate an f-cut for the integer variable whose fractional part is closest to 0.5.

(2) Resolve the LP using a dual simplex procedure.

In the computational results reported later we introduced up to 25 f-cuts and in the event that this procedure did not terminate with the optimal integer solution we adopted the following cutting plane rounding procedure:

(1) Choose the integer pattern use variable (e.g. x_{pi}) with the largest fractional part and add to the LP the cutting plane

$$x_{pi} \geqslant \lfloor X_{pi} \rfloor . \tag{22}$$

(2) Resolve the LP using a dual simplex procedure and repeat until an integer solution is obtained.

At the end of this rounding procedure it may again be possible to reduce down certain X_{pi} values (as mentioned in the last section) and we adopted the same approach as given previously to do this.

Examination of computational results showed that although (in general) cutting plane rounding produced a better integer solution than the simple rounding scheme, occasionally it was much worse, and so we implemented both rounding schemes and chose the best integer solution produced.

5.2. Stock rectangle selection

We present an interchange procedure for stock rectangle selection.

(1) Solve the cutting pattern selection problem with $K = [1, 2, \ldots, n]$, i.e. all stock rectangles available.

(2) Let U_i be the number of times a stock rectangle of type i is used in the solution to the cutting pattern selection problem. Without loss of generality renumber the stock rectangles such that $U_1 \geqslant U_2 \geqslant U_3 \geqslant \cdots \geqslant U_n$ (ties broken arbitrarily).

(3) Define $K = [1, 2, 3, \ldots, k-1, g]$ where $U_g = \max(U_i \mid i \geqslant k$ and each piece (l_j, w_j), $j = 1, \ldots, m$, can be cut from at least one of (L_1, W_1), $(L_2, W_2), \ldots, (L_{k-1}, W_{k-1})$, $(L_i, W_i))$ — we assume that g exists, if not the problem is infeasible. This definition ensures that the set K can feasibly cut all pieces (l_j, w_j), $j = 1, \ldots, m$, and note here that if $g = k$ and $U_{k+1} = 0$ we are finished.

(4) Define $S_T = [i \mid U_i$ one of the T highest U values, $i \notin K]$ (some $T \leqslant n - k$).

(5) We now consider all interchanges of some stock rectangle from K with some stock rectangle from S_T:

(a) For all pairs (i_1, i_2), $i_1 \in K$, $i_2 \in S_T$:

(b) If $K - [i_1] + [i_2]$ is a better (as evaluated by the heuristic for cutting pattern selection) stock rectangle set than K then put $K = K - [i_1] + [i_2]$ and $S_T = S_T - [i_2] + [i_1]$.

(c) The procedure terminates when no pair (i_1, i_2), $i_1 \in K$, $i_2 \in S_T$ can be found whose interchange leads to an improved solution (or when some limit on computation time is reached).

6. Computational results

The algorithm was programmed in FORTRAN and run on a CDC 7600 using the FTN compiler

Table 1
Computational results

Problem number	Number of pieces m	k	T	(L_0, W_0)	Number of interchanges	Number of pair examinations	Waste area percentage	Total time CDC 7600 (seconds)
1	10	2	2	(100, 100)	1	4	7.69	14.8
2	20				1	4	4.17	41.1
3	30				2	5	5.87	91.4
4	10	3	3	(100, 100)	2	11	6.63	26.5
5	20				2	13	4.95	116.7
6	30				1	9	7.62	313.3
7	10	2	2	(250, 250)	0	3	16.84	12.5
8	20				0	3	5.48	58.9
9	30				2	6	9.07	116.9
10	10	3	3	(250, 250)	1	9	13.80	24.4
11	20				4	22	6.65	204.3
12	30				1	10	5.89	182.3

with maximum optimisation for a number of randomly generated problems.

These problems were produced by choosing two values L_0, W_0 representing the maximum length and width of any stock rectangle. We then randomly generated n stock rectangles by sampling integers from the uniform distribution $[L_0/2, L_0]$ for stock rectangle lengths and by sampling integers from the uniform distribution $[W_0/2, W_0]$ for stock rectangle widths. The m pieces to be cut from these stock rectangles were generated by sampling integers from the uniform distribution $[L_0/4, L_0/2]$ for piece lengths and by sampling integers from the uniform distribution $[W_0/4, W_0/2]$ for piece widths.

The limits (a_j, b_j) on the number of pieces that we cut were produced by setting $a_j = 20$ $(j = 1,\ldots,m)$ with the b_j $(j = 1,\ldots,m)$ being generated by sampling integers from the uniform distribution [20,40]. All the problems were trim-loss minimisation problems with n (the number of stock rectangles) being equal to 10.

Table 1 gives the results for the problems solved. In that table we give for each problem, the number of interchanges and the number of stock rectangle pair examinations in the stock rectangle selection procedure, the total stock rectangle waste area as a percentage of the total stock rectangle area used and the total computation time in CDC 7600 seconds. Note here that the linear program was solved by in-core primal simplex and dual simplex procedures and also that (approximately) eighty per cent of the computation time shown was consumed in solving unconstrained two-dimensional guillotine cutting problems.

Looking at the waste area percentage we can see that (in general) the algorithm produces a relatively low amount of waste area although (as is common with heuristic procedures) because we do not know what the optimal solution is we cannot be sure how far from optimal the results obtained are. In order that our algorithm can be compared with those of other workers we present in Table 2 the details of problem 12 (the largest problem that we solved).

It is clear that, as part of our overall algorithm for the two-dimensional guillotine cutting assortment problem, we have developed a quite sophisticated cutting pattern selection algorithm. In order to test this part of the overall algorithm we solved the two-dimensional guillotine cutting problem given by Wang [12] taken from Skalbech and Schultz [10]. Our algorithm produced a solution requiring 327 312 square inches of stock of which 2.05% was waste in 8.3 seconds on a CDC 7600. This compares well with the result obtained by Wang [12] which required 326 736 square inches of stock of which 1.9% was waste obtained in approximately 13 minutes on a UNIVAC 1100/81 and the result of Skalbeck and Schultz [10] requiring 331 344 square inches of stock of which 3.2% was waste.

Table 2
Problem details

Stock rectangle	Length	Width	Piece	Length	Width	b_i	Piece	Length	Width	b_i
1	132	165	1	118	92	29	16	67	124	38
2	132	246	2	75	90	37	17	91	75	34
3	151	164	3	123	103	22	18	106	86	33
4	136	215	4	73	108	39	19	102	78	28
5	141	184	5	86	63	39	20	115	122	32
6	196	144	6	93	68	27	21	80	87	35
7	187	159	7	82	111	35	22	111	82	35
8	131	161	8	74	92	35	23	124	71	21
9	236	193	9	85	70	25	24	71	82	29
10	207	203	10	96	80	29	25	88	94	33
			11	120	105	33	26	74	102	37
			12	119	75	33	27	105	89	20
			13	73	103	20	28	82	121	30
			14	80	121	39	29	110	80	34
			15	78	105	24	30	82	115	23

Note: orientations of all pieces fixed (no rotations allowed), final objective function value 342 108, stock set [2,9,10].

7. Conclusions

In this paper we have presented a heuristic algorithm for the two-dimensional guillotine cutting assortment problem based upon a sophisticated cutting pattern generation procedure, a linear program and an interchange procedure. Computational results indicated that the algorithm developed is capable of dealing with moderate sized two-dimensional guillotine cutting assortment problems and producing good quality results. In addition the algorithm appears able to produce good results for the two-dimensional guillotine cutting problem.

References

[1] Abel, D., Dyckhoff, H., Gal, T. and Kruse, H.J., "Classification of real cutting stock problems", Discussion Paper 66a, Fernuniversitat, 5800 Hagen, 1983.

[2] Beasley, J.E., "An exact two-dimensional non-guillotine cutting tree search procedure", to appear in *Operations Research.*

[3] Chambers, M.L. and Dyson, R.G., "The cutting stock problem in the flat glass industry—selection of stock sizes", *Operational Research Quarterly* 27 (1976) 949–957.

[4] Dyckhoff, H., "A new linear programming approach to the cutting stock problem", *Operations Research* 29 (1981) 1092–1104.

[5] Gilmore, P.C. and Gomory, R.E., "A linear programming approach to the cutting-stock problem", *Operations Research* 11 (1963) 863–888.

[6] Gilmore, P.C. and Gomory, R.E., "The theory and computation of knapsack functions", *Operations Research* 14 (1966) 1045–1074.

[7] Hinxman, A.I., "The trim-loss and assortment problems: A survey", *European Journal of Operational Research* 5 (1980) 8–18.

[8] Page. E., "A note on a two-dimensional dynamic programming problem", *Operational Research Quarterly* 26 (1975) 321–324.

[9] Scarborough, S.G., "The single dimensional cutting problem", M.Sc. Thesis, Department of Management Science, Imperial College, London, SW7 2BX, England, 1982.

[10] Skalbeck, B.A. and Schultz, H.K., "Reducing trim waste in panel cutting using integer and linear programming", *Proceedings of the Western AIDS Conference,* March 1976.

[11] Taha, H.A., "*Integer programming: Theory, Applications and Computations*", Academic Press, New York–London, 1975.

[12] Wang, P.Y., "Two algorithms for constrained two-dimensional cutting stock problems", *Operations Research* 31 (1983) 573–586.