Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure

J. E. Beasley,

To cite this article:
J. E. Beasley, (1985) An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure. Operations Research 33(1):49-64. http://dx.doi.org/10.1287/opre.33.1.49

An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure

J. E. BEASLEY

Imperial College of Science and Technology, London, England

(Received May 1982; accepted October 1983)

We consider the two-dimensional cutting problem of cutting a number of rectangular pieces from a single large rectangle so as to maximize the value of the pieces cut. We develop a Lagrangean relaxation of a zero-one integer programming formulation of the problem and use it as a bound in a tree search procedure. Subgradient optimization is used to optimize the bound derived from the Lagrangean relaxation. Problem reduction tests derived from both the original problem and the Lagrangean relaxation are given. Incorporating the bound and the reduction tests into a tree search procedure enables moderately sized problems to be solved.

IN THE two-dimensional cutting problem, a single plane rectangular piece (called the stock piece) is to be cut into a number of smaller rectangular pieces, of given size and each with a given value. The objective is to maximize the value of the final product subject to the constraint that the number of cut pieces of the same size must lie within prescribed limits.

This problem arises in the cutting of steel or glass plates into required sizes, in the cutting of wood sheets to make furniture, the cutting of cardboard into boxes, and in the placing of advertisements on the pages of newspapers and magazines. The problem of minimizing the amount of waste produced can be converted into this problem by making the value of all pieces proportional to their areas.

A special case of this two-dimensional cutting problem restricts any cuts to be guillotine cuts—a guillotine cut on a rectangle is a cut from one edge of the rectangle to the opposite edge, parallel to the two remaining edges.

A number of authors have considered guillotine cutting problems. The unconstrained problem (where there is no constraint imposed upon the number of cut pieces of the same size that are produced) has been solved by dynamic programming (Gilmore and Gomory [1965, 1966]) and by the use of a recursive (tree search) procedure (Herz [1972]). Christofides and

Subject classification 582 and 627 two-dimensional cutting.

Operations Research
Vol 33, No. 1, January–February 1985

0030-364X/85/3301-0049 $01.25

Whitlock [1977] have presented a tree search procedure for the constrained guillotine cutting problem. Researchers (e.g., Wang [1983]) have also developed heuristic approaches to the guillotine cutting problem. Madsen [1980] lists references covering the problem.

In this paper we consider a version of the two-dimensional cutting problem in which the cuts are not restricted to be guillotine cuts. Non-guillotine cutting problems have been considered by relatively few authors in the literature.

The problem of packing rectangles into a unit width, infinite height, bin so as to minimize the total height of the packing has been studied by Baker et al. [1980, 1981] and by Coffman et al. [1980] who develop heuristic algorithms with guaranteed worst-case performance bounds. Biro and Boros [1984] have presented a heuristic algorithm for this problem based upon a network flow approach to generating feasible packings.

Smith and de Cani [1980], Bischoff and Dowsland [1982], and Dowsland [1982] have developed heuristic procedures for the pallet loading problem, which is a special case of the non-guillotine cutting problem with only one size of rectangular piece to be cut and with the objective of maximizing the number of cut pieces.

In this paper we present an algorithm for the two-dimensional non-guillotine cutting problem. As far as we are aware, no other exact solution procedure for this problem exists in the literature.

1. PROBLEM FORMULATION

In this section we formulate the two-dimensional cutting problem as a zero-one integer programming problem and consider how the formulation can be extended to deal with multiple stock rectangles, with defects in the stock rectangle, and with nonrectangular pieces.

1.1. Formulation

A large stock rectangle $A_0 = (L_0, W_0)$ of length L_0 and width W_0 is to be cut into m smaller rectangular pieces; piece i has size (L_i, W_i) and value v_i. Let P_i and Q_i be the minimum and maximum number of pieces of type i that can be cut from $A_0 (0 \leq P_i \leq Q_i$ for $i = 1, \cdots, m)$.

We assume

(i) L_0, W_0 and L_i, W_i for $i = 1, \cdots, m$ are integers.

(ii) The pieces are to be cut from A_0 so that their edges are parallel to the edges of A_0 (de Cani [1978] has relaxed this assumption).

(iii) The orientation of the pieces is fixed (i.e., a piece of length p and width q is not the same as a piece of length q and width $p (p \neq q)$). Note here that this assumption is not critical since it is a simple matter to extend the problem formulation and solution algorithm

given in this paper to cope with any nonsquare pieces that can be oriented in either direction.

We can then formulate the problem as follows. Let

$a_{ipqrs} = 1$ if a piece of type i, when cut with its bottom left-hand corner at (p, q), cuts out the point (r, s) (see Figure 1)
$= 0$ otherwise.

To prevent double counting when two pieces are cut adjacent to one another, we define

$a_{ipqrs} = 1$ if $0 \le p \le r \le p + L_i - 1 \le L_0 - 1$ and $0 \le q \le s \le q + W_i - 1 \le W_0 - 1$
$= 0$ otherwise;

i.e., we do not regard the top edge and the right-hand edge of a piece as cutting out points. Note that in order to define a_{ipqrs}, we must know the orientation of the cut pieces with respect to A_0.

Let $L = [0, 1, 2, \cdots, L_0 - 1]$ be the set of achievable lengths for the locations of the bottom left-hand corners of cut pieces and let $W = [0, 1, 2, \cdots, W_0 - 1]$ be the corresponding set for the widths.

Define

$x_{ipq} = 1$ if a piece of type i is cut with its bottom left-hand corner at (p, q) where $0 \le p \le L_0 - L_i$ and $0 \le q \le W_0 - W_i$ to ensure that it can be cut
$= 0$ otherwise.

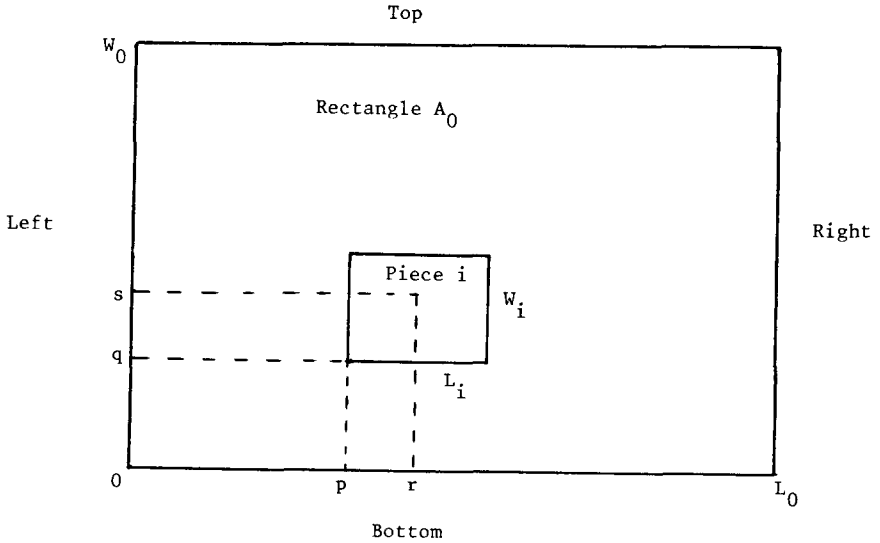

Figure 1. Locating a cut piece.

Then the program is

maximize

$$\sum_{i=1}^{m} \sum_{p \in L} \sum_{q \in W} v_i x_{ipq} \tag{1}$$

subject to

$$\sum_{i=1}^{m} \sum_{p \in L} \sum_{q \in W} a_{ipqrs} x_{ipq} \leq 1 \tag{2}$$

$$\text{for all} \quad r \in L, \quad \text{for all} \quad s \in W$$

$$P_i \leq \sum_{p \in L} \sum_{q \in W} x_{ipq} \leq Q_i, \; i = 1, \cdots, m \tag{3}$$

$$x_{ipq} \in (0, 1) \tag{4}$$

$$i = 1, \cdots, m \quad \text{for all} \quad p \in L, \quad \text{for all} \quad q \in W.$$

Constraint (2) ensures that any point is cut out by at most one piece, constraint (3) that the number of cut pieces of any type lies within the required range, and constraint (4) is the integrality constraint.

This model is a large zero-one integer program involving $0(m \mid L \parallel W \mid)$ variables and $0(\mid L \parallel W \mid)$ constraints. However, the size of this program can be reduced by restricting attention to normal cutting patterns as discussed below.

1.2. Normal Patterns

Normal patterns as used by Herz [1972] (who called them canonical dissections) and Christofides and Whitlock [1977] rely on the observation that, in a given cutting pattern, any cut piece can be moved to the left and/or down until its left-hand edge and its bottom edge are both adjacent to other cut pieces (or A_0). Such normal patterns permit us, without loss of optimality, to restrict the sets L and W. For example, the set $L = [0, 1, 2, \cdots, L_0 - 1]$ can be replaced by the set

$$L = [y \mid y = \sum_{i=1}^{m} L_i b_i, \; 0 \leq y \leq L_0 - \min(L_i \mid i = 1, \cdots, m) \cdot b_i \geq 0 \quad \text{and integer} \quad i = 1, \cdots, m]. \tag{5}$$

A similar definition applies for W. Christofides and Whitlock give the details of how to calculate these sets L and W by means of a simple dynamic programming recursion, which will not be repeated here.

Note that replacing the complete sets $L = [0, 1, 2, \cdots, L_0 - 1]$ and $W = [0, 1, 2, \cdots, W_0 - 1]$ by these restricted sets does not invalidate the formulation of the problem since it is easy to show that constraint (2) need be applied only for the restricted sets, and not the complete sets.

This use of normal patterns significantly reduces the size of the zero-one integer programming problem (Equations 1–4). For example, if A_0 is 2 m square and the pieces to be cut from A_0 are measured to an accuracy

of 1 mm, then the complete sets would have $|L| = |W| = 2000$. However, if the pieces to be cut are relatively large compared to A_0, then it is clear from Equation 5 that the restricted sets will be considerably smaller.

1.3. Multiple Stock Rectangles

Multiple stock rectangles can be dealt with by joining them together (in an arbitrary fashion) to form a large "superstock" rectangle. We illustrate the approach in Figure 2 where two stock rectangles each of size (L_0, W_0) are joined together to form a superstock rectangle of size $(2L_0, W_0)$.

Let the sets $L(W)$ of achievable lengths (widths) for the locations of bottom left-hand corners of cut pieces be specified as above for the single stock rectangle A_0 (of size (L_0, W_0)). Then for the superstock rectangle shown in Figure 2, the corresponding sets are $[p, p + L_0 | p \in L]$ for achievable lengths and W for achievable widths. The formulation of the non-guillotine cutting problem for the multiple stock rectangle case shown in Figure 2 is then the same as the formulation of the non-guillotine cutting problem for the single superstock rectangle with these modified sets of achievable lengths and widths, and with a_{ipqrs} and x_{ipq} redefined (in an obvious way) to ensure that pieces are not cut across the junction of the two stock rectangles.

1.4. Defects

Defects in the rectangle A_0 (as discussed by Hahn [1968] for guillotine cutting) are easy to accommodate in the model. If D represents the set of defective points that cannot be cut out by any piece, then we add the following constraint to the formulation:

$$\sum_{i=1}^{m} \sum_{p \in L} \sum_{q \in W} a_{ipqrs} x_{ipq} = 0, \quad \text{for all} \quad (r, s) \in D.$$

Note that the presence of defects does complicate the definition of a normal pattern—any cut piece must now be adjacent at its left-hand edge and bottom edge to another cut piece, or A_0, or to a point in D.

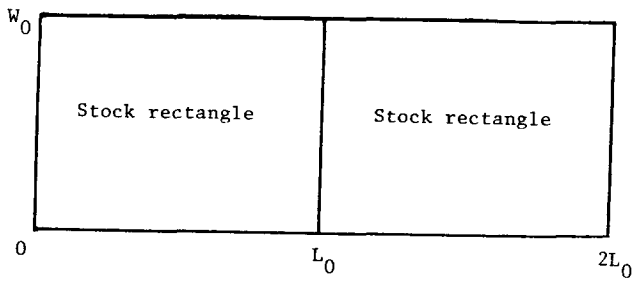

Figure 2. Combining multiple stocks.

1.5. Nonrectangular Pieces

The formulation of the problem can be extended to problems with nonrectangular pieces (for A_0 and/or for the pieces being cut) provided that a_{ipqrs} can still be defined. In these situations, many of the ideas contained in this paper apply. To see this, regard the formulation (Equations 1–4) as the non-guillotine cutting problem for the stock piece consisting of the set of points $[(r, s) \mid r \in L, s \in W]$ with the effect of cutting any piece i defined in terms of a reference point (p, q) (the bottom left-hand corner) and (a_{ipqrs}) that represents the points cut out by the piece when placed at the reference point.

2. THE UPPER BOUND

In this section we discuss how we can use the formulation of the non-guillotine cutting problem to solve the problem optimally. The solution procedure uses an upper bound derived from a Lagrangean relaxation of the formulation, in a tree search procedure.

2.1. Overview

The procedure that we have developed for solving the two-dimensional cutting problem can be viewed as a three stage method:

(a) A Lagrangean relaxation of the formulation of the problem provides an upper bound upon the optimal objective value,
(b) A subgradient optimization method attempts to minimize the upper bound from the Lagrangean relaxation,
(c) A tree search procedure resolves the problem.

Fisher [1981] discussed the use of Lagrangean relaxation in procedures of this type and indicated that they have been successful computationally. Our personal experience has been that the general approach given above has been effective when applied to location problems (Christofides and Beasley [1982, 1983]) and the set covering problem (Beasley [1983]).

In the next section we present the Lagrangean relaxation of the problem.

2.2. Lagrangean Relaxation

From constraint (2) derive the two constraints shown below by summing over all achievable lengths $r \in L$ (constraint (6)), and all achievable widths $s \in W$ (constraint (7))

$$\sum_{r \in L} \left(\sum_{i=1}^{m} \sum_{p \in L} \sum_{q \in W} a_{ipqrs} x_{ipq} \right) \leq |L| \quad \text{for all} \quad s \in W \qquad (6)$$

$$\sum_{s \in W} \left(\sum_{i=1}^{m} \sum_{p \in L} \sum_{q \in W} a_{ipqrs} x_{ipq} \right) \leq |W| \quad \text{for all} \quad r \in L. \qquad (7)$$

Introducing Lagrange multipliers $g_s(\geq 0)$ for all $s \in W$ for constraint (6) and $h_r(\geq 0)$ for all $r \in L$ for constraint (7), we obtain the Lagrangean program

maximize

$$\sum_{i=1}^{m} \sum_{p\in L} \sum_{q\in W} V_{ipq} x_{ipq} + \left(\sum_{s\in W} g_s\right) |L| + \left(\sum_{r\in L} h_r\right) |W|$$

subject to

$$P_i \leq \sum_{p\in L} \sum_{q\in W} x_{ipq} \leq Q_i \quad i = 1, \cdots, m$$

$$x_{ipq} \in (0, 1) \qquad i = 1, \cdots, m \text{ for all } p \in L, \text{ for all } q \in W$$

where $V_{ipq} = v_i - \sum_{r\in L} \sum_{s\in W} (g_s + h_r)\, a_{ipqrs}$. Note that the optimal value of this program for any set of nonnegative Lagrange multipliers is an upper bound on the optimal objective value of the original two-dimensional cutting problem.

This Lagrangean program can be easily solved, as it decomposes into m separate problems—one for each piece. Consider a piece of type i; then the terms in the Lagrangean program that relate to this piece can be extracted from that program to form the subproblem

maximize

$$\sum_{p\in L} \sum_{q\in W} V_{ipq} x_{ipq}$$

subject to

$$P_i \leq \sum_{p\in L} \sum_{q\in W} x_{ipq} \leq Q_i \tag{8}$$

$$x_{ipq} \in (0, 1) \qquad \text{for all} \quad p \in L, \quad \text{for all} \quad q \in W.$$

This subproblem corresponds to picking the best positions for the cutting out of between P_i and Q_i pieces of type i, where the effect of the Lagrangean relaxation has been to remove from the problem any restriction that the pieces cut should not overlap.

The subproblem can be solved by inspection. Its solution sets the x_{ipq} with the P_i largest V_{ipq} values to one and then considers the remaining x_{ipq} in decreasing V_{ipq} order and sets at most $(Q_i - P_i)$ of these to one provided that their V_{ipq} values are nonnegative.

Let (X_{ipq}) represent the corresponding values of the (x_{ipq}) in the solution of the Lagrangean program; then the optimal value of the Lagrangean program Z_{UB} as given by

$$Z_{UB} = \sum_{i=1}^{m} \sum_{p\in L} \sum_{q\in W} V_{ipq} X_{ipq} + \left(\sum_{s\in W} g_s\right) |L| + \left(\sum_{r\in L} h_r\right) |W|$$

is an upper bound on the optimal objective value of the original two-dimensional cutting problem.

3. THE SUBGRADIENT PROCEDURE

Subgradient optimization was used in an attempt to minimize the upper bound obtained from the Lagrangean relaxation of the problem. This technique has been discussed fairly widely in the literature (see Held et al. [1974], Sandi [1979], and Fisher [1981]). We adopted the following procedure:

1. Set $g_s = 0$ for all $s \in W$ and $h_r = 0$ for all $r \in L$ as initial values for the multipliers.
2. Solve the Lagrangean program with the current set of multipliers obtaining the solution (X_{ipq}) and objective value Z_{UB}.
3. If the Lagrangean solution (X_{ipq}) is a feasible solution to the original problem, then update Z_{LB}, the lower bound on the problem corresponding to a feasible solution, accordingly. Update the minimum upper bound $(Z_{\min})$ with Z_{UB}.
4. Stop if $Z_{\min} = Z_{LB}$—else go to Step 5.
5. Calculate the subgradients

$$G_s = -|L| + \sum_{r \in L} \left(\sum_{i=1}^{m} \sum_{p \in L} \sum_{q \in W} a_{ipqrs} X_{ipq}\right) \qquad \text{for all} \quad s \in W$$

$$H_r = -|W| + \sum_{s \in W} \left(\sum_{i=1}^{m} \sum_{p \in L} \sum_{q \in W} a_{ipqrs} X_{ipq}\right) \qquad \text{for all} \quad r \in L.$$

6. Define a step size t by

$$t = f(Z_{UB} - Z_{LB}) / \left(\sum_{s \in W} (G_s)^2 + \sum_{r \in L} (H_r)^2\right), \tag{9}$$

where $0 < f \leq 2$, and update the multipliers by

$$g_s := \max(0, g_s + tG_s) \qquad \text{for all} \quad s \in W$$

$$h_r := \max(0, h_r + tH_r) \qquad \text{for all} \quad r \in L.$$

7. Go to Step 2 to resolve the Lagrangean program with this new set of multipliers unless sufficient subgradient iterations have been performed, in which case stop.

In calculating a value for f (Equation 9), we followed the approach of Held et al. [1974] in letting $f = 2$ for $2(|L| + |W|)$ iterations, then successively halving both f and the number of iterations until the number of iterations reached a threshold value of five; f was then halved every five iterations. We terminated the subgradient procedure when f fell below 0.005.

At the end of this subgradient procedure, the optimal solution to the original two-dimensional cutting problem may have been found (if we had $Z_{\min} = Z_{LB}$ at Step 4 above), but if not, we resolve the problem using a tree search procedure. Before describing this tree search procedure, however, we discuss the problem reduction tests that can be applied to reduce the size of the problem that we need to solve.

4. PROBLEM REDUCTION

There are a number of reduction tests that can be used to reduce the size of the problem. In this section we outline those that we used.

(1) Overlapping Pieces

Suppose that we have two pieces, (i and j say), that overlap such that $L_i + L_j > L_0$; i.e., they cannot both be cut with their bottom left-hand corners at the same width. Then we can update Q_i, the maximum number of pieces of type i that we can cut out of A_0, by

$$Q_i := \min(Q_i, \lfloor L_0/L_i \rfloor \lfloor (W_0 - \lceil P_j/\lfloor L_0/L_j \rfloor \rceil W_j)/W_i \rfloor), \qquad (10)$$

where $\lfloor y \rfloor$ denotes the largest integer less than or equal to y and $\lceil y \rceil$ denotes the smallest integer greater than or equal to y. Equation 10 is derived by considering how much of A_0 is taken up with cutting out P_j pieces of type j and using the remainder of A_0 to cut out pieces of type i. An expression like (10) holds for Q_j and also for pieces for which $W_i + W_j > W_0$.

(2) Free Area

As we must use an area of at least $\sum_{j=1}^{m} P_j L_j W_j$ in cutting pieces out of A_0, we have

$$Q_i := \min(Q_i, P_i + \lfloor (L_0 W_0 - \textstyle\sum_{j=1}^{m} P_j L_j W_j)/(L_i W_i) \rfloor).$$

(3) Free Value

Given a value Z_{UB} corresponding to an upper bound on the optimal solution to the original two-dimensional cutting problem, then since $\sum_{j=1}^{m} P_j v_j$ is accounted for, we have that

$$Q_i := \min(Q_i, P_i + \lfloor (Z_{UB} - \textstyle\sum_{j=1}^{m} P_j v_j)/v_i \rfloor).$$

(4) Penalties on Number of Cut Pieces

It is clear from the structure of the Lagrangean program that we can calculate an upper bound on the solution obtained with exactly d pieces of type i cut from A_0. Let U_{ij} represent the jth largest V_{ipq} ($p \in L$, $q \in W$) value; then the upper bound with exactly d pieces of type i cut from A_0 is given by

$$Z_{UB} - \textstyle\sum_{p \in L} \sum_{q \in W} V_{ipq} X_{ipq} + \sum_{j=1}^{d} U_{ij}.$$

By investigating all values of d where $P_i \le d \le Q_i$ and comparing the upper bounds obtained with Z_{LB}, a lower bound on the problem corresponding to a feasible solution, we can update P_i and Q_i accordingly; e.g.,

if the upper bound obtained when $d = Q_i$ is less than Z_{LB}, then we can reduce Q_i by one.

(5) Penalties on Cut Positions

It is clear from the structure of the Lagrangean program that penalties can be calculated for the setting of x_{ipq} to one (or zero), i.e., cutting a piece of type i with its bottom left-hand corner at (p, q) (or not). For example, the upper bound obtained when setting x_{ipq} to one (where $X_{ipq} = 0$) is given by:

$$Z_{UB} + V_{ipq} - \min_{r\in L, s\in W, X_{irs}=1}(V_{irs}) \quad \text{if} \quad \sum_{r\in L}\sum_{s\in W} X_{irs} = Q_i \quad (11)$$

$$Z_{UB} + V_{ipq} - \min_{r\in L, s\in W, X_{irs}=1}(0, (V_{irs})) \quad \text{if} \quad \sum_{r\in L}\sum_{s\in W} X_{irs} \neq Q_i. \quad (12)$$

These penalties preserve the condition (Equation 8) that the number of pieces of type i that are cut from A_0 lies between P_i and Q_i. If the penalty value (Equations 11 and 12) is less than Z_{LB}, then we cannot set x_{ipq} to one in the optimal solution and so x_{ipq} can be deleted from the problem.

(6) Area Program

Define y_i as the number of pieces of type i cut from A_0; then the program

maximize

$$\sum_{i=1}^{m} v_i y_i$$

subject to

$$P_i \leq y_i \leq Q_i \qquad i = 1, \cdots, m$$

$$\sum_{i=1}^{m} L_i W_i y_i \leq L_0 W_0$$

$$y_i \text{ integer} \qquad i = 1, \cdots, m,$$

which limits the area of the pieces cut from A_0, is clearly an upper bound on the optimal solution to the original two-dimensional cutting problem. This program can be viewed as a knapsack problem and is easily solved (e.g., by the standard dynamic programming algorithm for the knapsack problem). Two reduction tests were derived from this program—these were

(a) A penalty value for cutting exactly d pieces of a particular type from A_0, and
(b) The same as (a), but based on the linear programming relaxation of the above program.

A similar program applies with the objective function coefficients being

Lagrangean values (V_{ipq}). As before, let U_{ij} represent the jth largest V_{ipq} value ($p \in L$, $q \in W$). Define $y_{ij} = 1$ if a piece i is cut corresponding to the jth largest V_{ipq} from A_0; $y_{ij} = 0$ otherwise, then the program

maximize

$$\sum_{i=1}^{m} \sum_{j=1}^{Q_i} U_{ij} y_{ij}$$

subject to

$$P_i \leq \sum_{j=1}^{Q_i} y_{ij} \qquad i = 1, \cdots, m$$

$$\sum_{i=1}^{m} \sum_{j=1}^{Q_i} L_i W_i y_{ij} \leq L_0 W_0$$

$$y_{ij} \in (0, 1) \qquad j = 1, \cdots, Q_i \quad i = 1, \cdots, m$$

is also an upper bound on the optimal solution to the original two-dimensional cutting problem. This program can also be viewed as a knapsack problem and can be used as discussed under (a) and (b) above to reduce the size of the problem.

5. THE TREE SEARCH PROCEDURE

As discussed previously, at the end of the subgradient procedure the minimum upper bound $Z_{\min}$ and the maximum lower bound Z_{LB} corresponding to a feasible solution may not coincide, in which case the problem has to be resolved. In this section, we discuss the tree search procedure we adopted to resolve the problem and we start by giving an overview of that procedure.

5.1. Tree Search Overview

We chose to resolve the problem using a binary, depth-first, tree search procedure forming a normalized cutting pattern in the tree and computing at each tree node an upper bound for the optimal completion of the node.

Each forward branch in the tree represented setting some x_{ipq} to one, i.e., choosing to cut a piece of type i with its bottom left-hand corner at (p, q), where (p, q) was chosen to ensure that the pattern of cut pieces formed a normal cutting pattern. We can regard the tree search as slowly building up a complete normalized cutting pattern for A_0.

It is clear from the Lagrangean program that we can calculate an upper bound for the optimal completion of each tree node as that program is easily adapted to cope with the setting of (x_{ipq}) to specific values in the tree. If this upper bound is less than Z_{LB}, then backtracking can occur.

A binary depth-first tree search procedure was chosen rather than, for example, a breadth-first tree search procedure, since it is easily implemented computationally; also, we had successfully used it before.

5.2. Heuristic Solution

It is clear that the performance of the tree search and reduction tests is dependent upon the quality of the lower bound Z_{LB} (corresponding to a feasible solution) on the optimal objective value. For this reason, we constructed a heuristic procedure that attempts to find a good feasible solution from any solution to the Lagrangean program. The procedure was as follows.

Let $F = [(i, p, q) \mid x_{ipq} = 1]$; i.e., F represents the set of pieces (and their cut positions) that must be cut from A_0. Typically, F consists of pieces that have been explicitly cut from A_0 in the tree.

1. Cut all pieces in F from A_0 (note F may be empty).
2. Form a list of $[(i, p, q) \mid X_{ipq} = 1$ and $(i, p, q) \notin F]$ arranged in descending V_{ipq} order.
3. Taking each piece in this list in turn, cut it out of A_0 if possible. Note that we do not require the cutting pattern formed to be a normalized cutting pattern.
4. At the end of Step 3 there will probably be some of A_0 that is not part of any cut piece; let $F^1 = [(r, s) \mid (r, s)$ not cut out of A_0, $r \in L$, $s \in W]$.
5. If F^1 is empty, go to Step 7; else go to Step 6.
6. Let $(R, S) = \min_{s \in W}(\min_{r \in L}((r, s) \mid (r, s) \in F^1))$. Cut out at this free point (R, S) the piece with the largest V_{iRS} value such that the number of pieces of type i cut from A_0 does not exceed Q_i; if no such piece exists, delete (R, S) from F^1 and go to Step 5; else update F^1 by removing from it the points cut out and go to Step 5.
7. If the solution constructed as above is feasible, then update Z_{LB} accordingly.

We now give the complete procedure for the problem. We indicate below how we structured the initial tree node and the tree search nodes with respect to the calculation of the bound and the reduction tests we carried out.

5.3. Initial Tree Node

(a) *Heuristic Solution*

We first calculated a heuristic solution to the problem using the procedure given above with all Lagrange multipliers at value zero, $L = [0, 1, \cdots, L_0 - 1]$ and $W = [0, 1, \ldots, W_0 - 1]$. This procedure was then repeated 50 times with (V_{ipq}) multiplied by a real random number drawn from the uniform distribution [0, 1]. The maximum value feasible solution found was used as an initial value for Z_{LB}.

(b) *Reduction*

We then carried out the reduction tests of Section 4 based on overlapping pieces and free area to reduce (Q_t) and then the area program reduction to update (P_t) and (Q_t).

(c) *Normal patterns*

The sets L and W of achievable lengths and widths were then calculated as outlined in Section 1.2.

(d) *Subgradient procedure*

The subgradient procedure was then carried out, with, at each subgradient iteration,

(i) The heuristic solution procedure being used to update Z_{LB}, and
(ii) The reduction tests of Section 4 based on overlapping pieces, free area, free value and the number of cut pieces were performed.

At the end of the subgradient procedure, the set of multipliers associated with the best upper bound found were recalled and the Lagrangean program resolved with that set of multipliers.

(e) *Reduction*

All the reduction tests outlined in Section 4 were then performed.

5.4. Other Tree Nodes

(a) *Reduction*

All the reduction tests outlined in Section 4 except the test relating to the area program (that requires the solution of a knapsack problem) were used at each tree node. The sets L and W were also recalculated at each tree node.

(b) *Bound*

At each tree node, we carried out five subgradient iterations with $f = 1$ (these parameters being set after exploring a number of possibilities). The initial set of multipliers at each tree node were the set associated with the best upper bound found at the predecessor tree node, and the initial (P_t), (Q_t) at each tree node were the final (P_t), (Q_t) at the predecessor tree node except that (P_t) was updated at each forward branch to take account of the pieces cut from A_0. The heuristic solution procedure was used at each subgradient iteration.

(c) *Branching*

At any tree node, let $G = [(i, p, q) \notin F \mid x_{ipq} \neq 0]$; i.e., G is the set of positions that can still have a piece cut out of them (and the corresponding pieces). Then define

$$S = \min(s \mid (i, r, s) \in G),\ R = \min(r \mid (i, r, S) \in G), \quad \text{and}$$

$$V_{jRS} = \max(V_{iRS} \mid (i, R, S) \in G).$$

We branched by cutting the piece j out of A_0 with its bottom left-hand corner at (R, S). Intuitively, this choice corresponds to cutting from A_0 the piece j with the maximum Lagrangean value that can fit in the lowest, then leftmost, point (R, S) so that we are forming a normalized cutting pattern in the tree. Note here that putting $x_{jRS} = 1$ means that $x_{ipq} = 0$ if there exists a point (r, s) such that $a_{jRSrs} + a_{ipqrs} = 2$.

(d) *Backtracking*

We can backtrack in the tree when any of the conditions given below are fulfilled:

(i) $\min(\sum_{i=1}^{m} Q_i v_i, Z_{UB}) \leq Z_{LB}$

(ii) $\sum_{i=1}^{m} P_i L_i W_i > L_0 W_0$

(iii) $\sum_{i=1}^{m} P_i v_i > Z_{UB}$.

6. COMPUTATIONAL RESULTS

The algorithm was programmed in FORTRAN and run on a CDC 7600 using the FTN compiler with maximum optimization for a number of randomly generated problems.

The random problems were produced by generating m real random numbers r_i for $i = 1, \cdots, m$ from the uniform distribution $(0, L_0 W_0/4)$. The dimension L_i of each piece was generated by sampling an integer from the uniform distribution $[1, L_0]$ and the dimension W_i by setting $W_i = \lceil r_i/L_i \rceil$. The integer value of each piece v_i was generated by multiplying $L_i W_i$ by a real random number drawn from the uniform distribution [1, 3] and rounding down, with $P_i = 0$ for $i = 1, \cdots, m$ and Q_i generated by sampling an integer from the uniform distribution [1, 3]. Table I gives details of the problems solved.

In Table I we give for each problem the size of the sets L and W together with the best upper bound ($Z_{\min}$) and also the best lower bound (Z_{LB} corresponding to a feasible solution) found at the initial tree node. To try and obtain a measure of the effectiveness of the reduction tests, let U_1 represent the value of $\sum_{i=1}^{m} (Q_i - P_i)$ at the start of the problem and U_2 the value of $\sum_{i=1}^{m} (Q_i - P_i)$ at the end of the initial tree node. Then, we give in Table I, for each problem, the reduction percentage

TABLE I

COMPUTATIONAL RESULTS

Problem number	(L_0,W_0)	m	|L|	|W|	Initial tree node: Reduction percentage	Initial tree node: Upper bound Z_{min}	Initial tree node: Lower bound Z_{LB}	Initial tree node: Time CDC 7600 seconds	Tree search: Optimum	Tree search: Number of tree nodes	Total time CDC 7600 seconds
1	(10,10)	5	7	6	100%	164	164	0.9	164	-	0 9
2		7	10	10	29%	247	230	3.4	230	4	4.0
3		10	9	10	38%	260	246	4.3	247	89	10.5
4	(15,10)	5	2	10	100%	268	268	0.1	268	-	0.1
5		7	6	10	100%	358	358	0.4	358	-	0.4
6		10	13	10	20%	317	289	7.6	289	355	55 2
7	(20,20)	5	10	20	100%	430	430	0.5	430	-	0 5
8		7	6	20	15%	915	834	6.9	834	2226	218.6
9		10	18	18	100%	930	924	18 2	924	2	18.3
10	(30,30)	5	6	16	100%	1452	1452	0 9	1452	-	0.9
11		7	18	27	13%	1860	1688	21 4	1688	226	79.1
12		10	27	30	9%	1982	1770	56.4	1865	453	229.0

$100(1 - U_2/U_1)$. The larger this value the greater the reduction that has been achieved.

The FORTRAN code for the problem required no special data structures with the storage requirement being $0(m \mid L \parallel W \mid)$, since we found it convenient to store the (V_{ipq}) values rather than recalculate them each time they were needed.

Overall, Table I indicates that the Lagrangean-based tree search algorithm is capable of solving moderately sized non-guillotine cutting problems.

REFERENCES

BAKER, B. S., E. G. COFFMAN AND R. L. RIVEST. 1980. Orthogonal Packings in Two Dimensions. *SIAM J. Comput.* **9,** 846–855.

BAKER, B. S., D. J. BROWN AND H. P. KATSEFF. 1981. A 5/4 Algorithm for Two-Dimensional Packing. *J. Algorithms* **2,** 348–368.

BEASLEY, J. E. 1983. An Algorithm for Set Covering Problems. Department of Management Science, Imperial College, London SW7 2BX, England.

BIRO, M., AND E. BOROS. 1984. Network Flows and Non-Guillotine Cutting Patterns. *Eur. J. Opnl. Res.* **16,** 215–221.

BISCHOFF, E., AND E. B. DOWSLAND. 1982. An Application of the Micro to Product Design and Distribution. *J. Opnl. Res. Soc.* **33,** 271–280.

CHRISTOFIDES, N., AND J. E. BEASLEY. 1982. A Tree Search Algorithm for the *p*-Median Problem. *Eur. J. Opnl. Res.* **10,** 196–204.

CHRISTOFIDES, N., AND J. E. BEASLEY. 1983. Extensions to a Lagrangean Relaxation Approach for the Capacitated Warehouse Location Problem. *Eur. J. Opnl. Res.* **12,** 19–28.

CHRISTOFIDES, N., AND C. WHITLOCK. 1977. An Algorithm for Two-Dimensional Cutting Problems. *Opns. Res.* **25,** 30–44.

COFFMAN, E. G., M. R. GAREY, D. S. JOHNSON AND R. E. TARJAN. 1980. Performance Bounds for Level-Orientated Two-Dimensional Packing Algorithms. *SIAM J. Comput.* **9,** 808–826.

DE CANI, P. 1978. A Note on the Two-Dimensional Rectangular Cutting-Stock Problem. *J. Opnl. Res. Soc.* **29,** 703–706.

DOWSLAND, K. A. 1982. Two-Dimensional Rectangular Packing. M.Sc. thesis, Department of Management Science, University of Wales, Swansea, Wales.

FISHER, M. L. 1981. The Lagrangean Relaxation Method for Solving Integer Programming Problems. *Mgmt. Sci.* **27,** 1–18.

GILMORE, P. C., AND R. E. GOMORY. 1965. Multistage Cutting Problems of Two and More Dimensions. *Opns. Res.* **13,** 94–120.

GILMORE, P. C., AND R. E. GOMORY. 1966. The Theory and Computation of Knapsack Functions. *Opns. Res.* **14,** 1045–1075.

HAHN, S. G. 1968. On the Optimal Cutting of Defective Sheets. *Opns. Res.* **16,** 1100–1114.

HELD, M., P. WOLFE AND H. P. CROWDER. 1974. Validation of Subgradient Optimisation. *Math. Prog.* **6,** 62–88.

HERZ, J. C. 1972. A Recursive Computing Procedure for Two-Dimensional Stock Cutting. *I.B.M. J. Res. Dev.* **16,** 462–469.

MADSEN, O. B. G. 1980. References Concerning the Cutting Stock Problem. IMSOR, The Technical University of Denmark, DK-2800, Lyngby, Denmark.

SANDI, C. 1979. Subgradient Optimisation. In *Combinatorial Optimisation*, ed. N. Christofides, A. Mingozzi, P. Toth and C. Sandi. John Wiley & Sons, New York.

SMITH, A., AND P. DE CANI. 1980. An Algorithm to Optimise the Layout of Boxes in Pallets. *J. Opnl. Res. Soc.* **31,** 573–578.

WANG, P. Y. 1983. Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems. *Opns. Res.* **31,** 573–586.