Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems
Author(s): P. Y. Wang
Source: *Operations Research*, Vol. 31, No. 3 (May - Jun., 1983), pp. 573-586
Published by: INFORMS
Stable URL: http://www.jstor.org/stable/170624
Accessed: 12/03/2009 15:16

Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems

P. Y. WANG

University of Maryland-Baltimore County, Catonsville, Maryland

(Received May 1981; accepted October 1982)

We propose two combinatoric methods that generate constrained cutting patterns by successive horizontal and vertical builds of ordered rectangles. Each of the algorithms uses a parameter to bound the maximum acceptable percentages of waste they create. Error bounds measure how close the pattern wastes are to the waste of the optimal solution. We also discuss computational results and applications of the methods to a general cutting stock problem.

THE CONSTRAINED two-dimensional cutting stock problem seeks an optimal cutting pattern that cuts a given set of rectangles from a single stock sheet with minimum waste. The problem bounds the maximum number of times a rectangle may appear in the solution. This problem is a simplification of a general type of rectangular cutting stock problem that occurs in the glass and lumber industries. Discussions of these problems may be found in Brown [1971], Dyson and Gregory [1974], or Skalbeck and Schultz [1976].

The constrained problem has been studied by Christofides and Whitlock [1977] who proposed a solution method that incorporates both dynamic programming and a transportation routine into a tree-search algorithm. Our solution methods will also assume that acceptable cutting patterns are limited to those of guillotine type. That is, the rectangles are to be obtained by successive edge-to-edge cuts of the stock sheet. Furthermore, we shall take an opposite approach to determining all feasible guillotine patterns. Instead of enumerating all possible cuts that can be made on the stock sheet, our combinatoric algorithms will find guillotine cutting patterns by successively adding the rectangles to each other. To reduce the number of such combinations, we employ parameters to reject undesirable additions. For a given choice of these parameters, our algorithm, and optimality conditions that we prove, determine error bounds that measure the closeness of the best patterns to the optimal solution. In addition, we present a computational example and discuss the use of our algorithms for solving a general stock cutting problem.

Subject classification: 582 algorithms for two-dimensional cutting stock problems.

Operations Research
Vol. 31, No. 3, May–June 1983

0030-364X/83/3103-0573 $01.25

1. DEFINITIONS

The constrained rectangular cutting stock problem can be stated as follows. Let $H \times W$ be a rectangular stock sheet having height H and width W, and let R be a set of rectangles $R_1, R_2, \cdots, R_n$ with dimensions $h_1 \times w_1$, $h_2 \times w_2$, $\cdots$, $h_n \times w_n$. Determine the guillotine pattern with minimum trim waste that cuts the rectangles using no more than b_i replicates of rectangle R_i for $i = 1, 2, \cdots, n$ in the pattern. The problem can also be stated in the form:

$$\begin{aligned} &\text{Maximize}_G \textstyle\sum_{i=1}^{n} x_i h_i w_i \\ &\text{subject to} \quad 0 \le x_i \le b_i \\ &\qquad\qquad\quad x_i \text{ integer} \end{aligned} \qquad (i = 1, 2, \cdots, n).$$

In this formulation, x_i is an integer indicating the number of times the rectangle R_i appears in a guillotine cutting pattern G. The mathematical program requires an additional constraint to ensure that H and W are not exceeded.

We define a *guillotine pattern* G that cuts a subset of the rectangles $R_1, R_2, \cdots, R_n$ from a stock rectangle $H \times W$ to be a pattern from which the rectangles can be obtained by sequential edge-to-edge cuts made parallel to the edges of the stock sheet. The total area of the regions that do not contain any rectangles of R will be referred to as the *trim waste* of the pattern. In addition, as discussed by Christofides and Whitlock, every guillotine pattern has an equivalent normalized guillotine form. In this form, all rectangles in a pattern are left-justified at the lowest possible position in the stock sheet and placed adjacently as in Figure 1.

For a given guillotine pattern G, the corresponding *guillotine rectangle* S is defined to be the rectangle that contains the rectangles R_i of G and has the smallest possible height and width dimensions. Figure 2 shows the guillotine rectangle corresponding to Figure 1. Furthermore, we shall consider as equivalent, two guillotine rectangles containing the same set of rectangles and having the same height and width dimensions.

The algorithms we subsequently describe will generate sets of cutting patterns as guillotine rectangles containing combinations of the rectangles of R. Our methods will employ the following terminology.

A *horizontal build* of two rectangles $A_1 = p_1 \times q_1$ and $A_2 = p_2 \times q_2$ is a rectangle S_u having dimensions $\max(p_1, p_2) \times (q_1 + q_2)$ and containing A_1 and A_2. A *vertical build* of A_1 and A_2 is a rectangle S_v of dimensions $(p_1 + p_2) \times \max(q_1, q_2)$ that contains A_1 and A_2. An example is given in Figure 3. We also require that the height and width dimensions of S_u and S_v do not exceed the corresponding dimensions of the stock rectangle.

We will use two parameters β_1 and β_2 to denote the maximum acceptable percentages of waste of any guillotine rectangle T generated in our algorithms. Our first algorithm will measure β_1 with respect to the

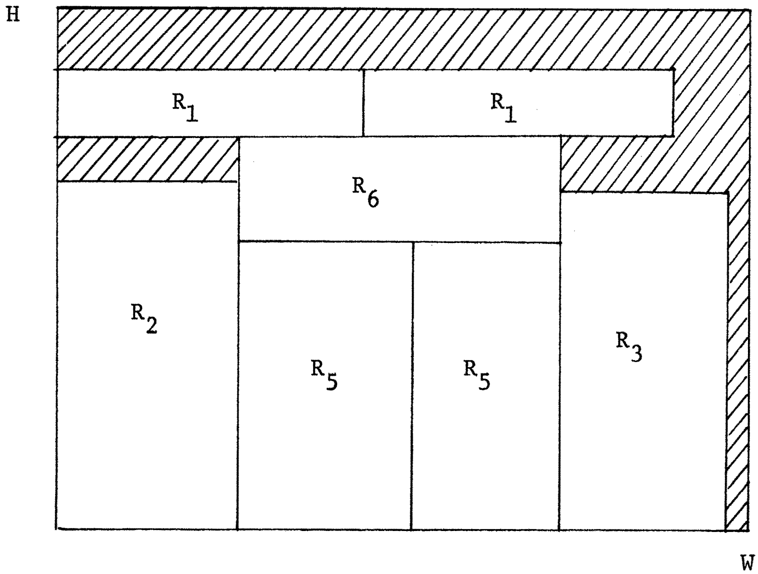

Figure 1. A normalized guillotine pattern G.

area of the stock sheet $H \times W$, and the second algorithm will measure β_2 with respect to the area $a(T)$ of T.

2. THE METHODS

The approach taken in our combinatoric procedures is to horizontally and vertically build together the rectangles $R_1, R_2, \cdots, R_n$ with each other. The algorithm then adds the resulting rectangles to the original

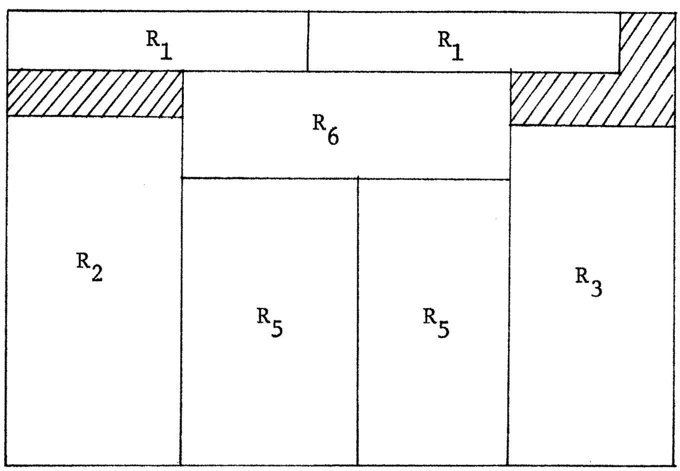

Figure 2. The guillotine rectangle, S, corresponding to G.

rectangles R_i to form a larger set of guillotine rectangles. The process is repeated so that each successive horizontal and vertical build forms a larger guillotine rectangle from two smaller guillotine rectangles. In addition, the two algorithms reject those constructed rectangles having percentages of waste exceeding either β_1 or β_2, respectively. Guillotine rectangles that contain more than b_i replicates of a rectangle R_i are also eliminated from further consideration. In this manner, the algorithm generates all possible rectangles subject to the constraints and parameters. Finally, the best rectangle is selected from this set. The two methods are described in more detail below.

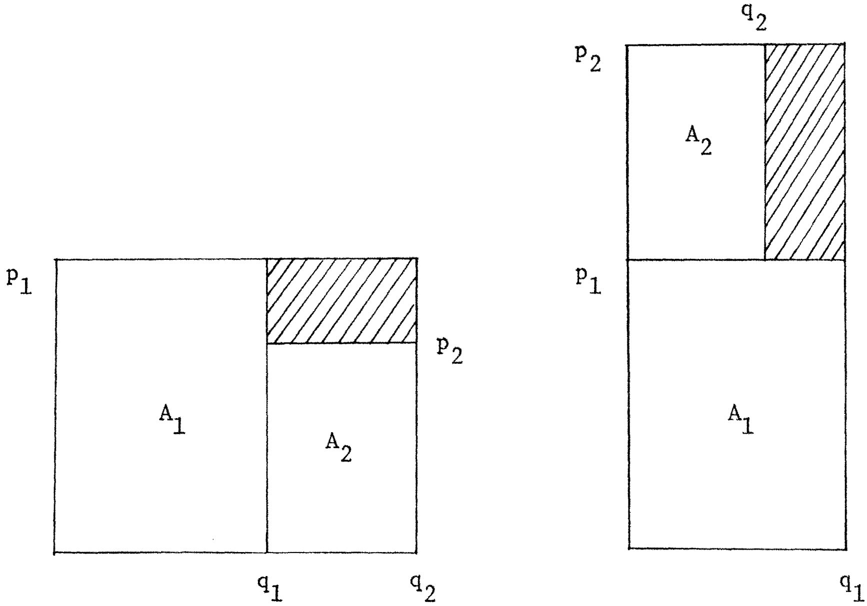

Figure 3. A horizontal and vertical build of A_1 and A_2.

Algorithm One

Step 1.a. Choose a value for β_1, $0 \leq \beta_1 \leq 1$.

b. Define $L^{(0)} = F^{(0)} = \{R_1, R_2, \cdots, R_n\}$, and set $k = 1$.

Step 2.a. Compute $F^{(k)}$ which is the set of all rectangle T satisfying

(i) T is formed by a horizontal or vertical build of two rectangles from $L^{(k-1)}$,

(ii) the amount of trim waste in T does not exceed $\beta_1 HW$, and

(iii) those rectangles R_i appearing in T do not violate the bound constraints $b_1, b_2, \cdots, b_n$.

b. Set $L^{(k)} = L^{(k-1)} \cup F^{(k)}$. Remove any equivalent rectangle pattterns from $L^{(k)}$.

Step 3. If $F^{(k)}$ is nonempty, set $k \leftarrow k + 1$ and go to Step 2. Otherwise,
*Step 4.*a. Set $M = k - 1$.
b. Choose the rectangle of $L^{(M)}$ that has the smallest total trim waste when placed in the stock sheet $H \times W$.

Algorithm Two

This procedure is a modification of Algorithm One. β_2 replaces β_1 in Step 1.a, and the following condition replaces Step 2.a (ii):
(ii) the amount of trim waste in T does not exceed $\beta_2 a(T)$.

3. ERROR BOUNDS AND OPTIMALITY CONDITIONS

As we have noted, the rectangles in $L^{(M)}$ of both algorithms are guillotine rectangles formed by a sequence of horizontal and vertical builds of the rectangles $R_1, R_2, \cdots, R_n$. The addition of two rectangles in this manner to form a larger rectangle is the reverse of guillotine cutting the larger rectangle into the two smaller ones. With β_1 or β_2 set equal to one, either algorithm will generate $L^{(M)}$ as the set of all possible guillotine rectangles that can be constructed from the R_i subject to the bound constraints. With β_1 or β_2 set equal to zero, $L^{(M)}$ will consist of rectangle patterns having no trim waste.

When the values of the parameters are increased from zero to one for a given constrained problem, the number of rectangles generated by reapplying the algorithms with these parameter values increases dramatically. With larger values of the β_i, better solutions are obtained since the size of $L^{(M)}$ increases. However, the algorithm then requires more computing time and storage space to generate each $L^{(k)}$. Thus, these algorithms will usually be employed with small values for β_1 and β_2. It is, therefore, advantageous to obtain upper bounds measuring the closeness of the optimal solution of the given problem to the best solutions obtained by the algorithms.

Let ω denote the trim waste in a rectangle $S = h \times w$ that was formed by a sequence of horizontal and vertical builds of $R_1, R_2, \cdots, R_n$ subject to the constraints $b_1, b_2, \cdots, b_n$ and the value of β_i. We refer to ω as the *inner waste*. Let $\mathscr{W}$ be the total trim waste associated with S when it is placed in the stock sheet $H \times W$ as in Figure 4. Then $\mathscr{W} = HW - hw + \omega$.

If $\mathscr{W}^*$ denotes the minimum amount of total trim waste that can be attained by any guillotine pattern which cuts $R_1, R_2, \cdots, R_n$ from the stock sheet without rotating the rectangles or violating the constraints, then we define $\mathscr{O}$ to be the set of all guillotine rectangles that have trim waste $\mathscr{W}^*$.

To obtain an error bound for Algorithm One, let $\beta_1{}^*$ be the smallest value of β_1 that allows the creation of at least one rectangle of $\mathscr{O}$. If $S_1{}^*$ is such a rectangle having inner waste $\omega_1{}^*$, then $\mathscr{W}^* \geq \omega_1{}^* = \beta_1{}^* HW$.

THEOREM 1. *If S is the guillotine rectangle of $L^{(M)}$ selected in Algorithm One as the pattern with minimum total waste $\mathscr{W}$, and the algorithm was applied with a specified value of β_1, then*

$$\mathscr{W} - \mathscr{W}^* \leq |\mathscr{W} - \beta_1 HW|.$$

Proof. We consider two cases.

(i) If $\beta_1 < \beta_1^*$, then S is not in $\mathscr{O}$. Hence $\mathscr{W} > \mathscr{W}^* \geq \beta_1^* HW > \beta_1 HW$ and $0 < \mathscr{W} - \mathscr{W}^* < \mathscr{W} - \beta_1 HW$.

(ii) If $\beta_1 \geq \beta_1^*$, then S is in $\mathscr{O}$ and $\mathscr{W} = \mathscr{W}^*$. This implies that $\mathscr{W} - \mathscr{W}^* = 0 \leq |\mathscr{W} - \beta_1 HW|$.

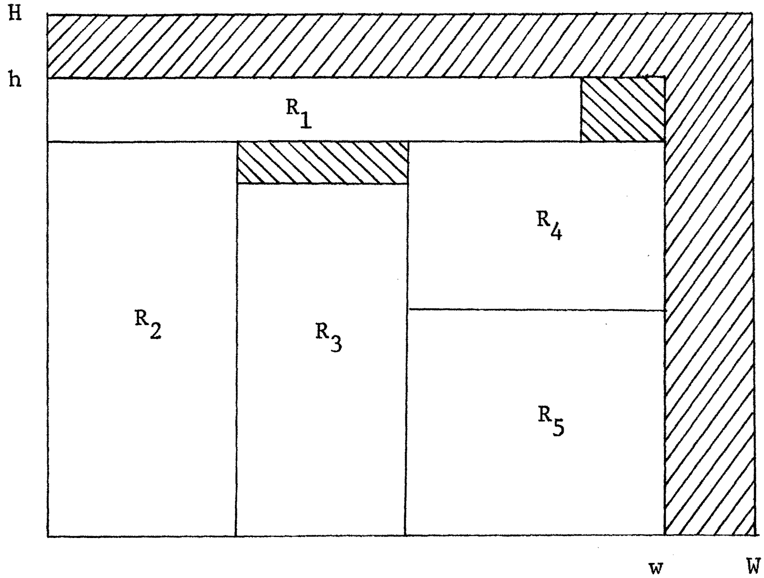

Figure 4. A guillotine cutting pattern generated by the algorithms.

THEOREM 2. *If the waste $\mathscr{W}$ of the pattern S obtained from Algorithm One with a fixed value of β_1 satisfies $\mathscr{W} \leq \beta_1 HW$, then S is an optimal pattern and $\mathscr{W} = \mathscr{W}^*$.*

Proof. Assume $\mathscr{W} \leq \beta_1 HW$. If $S \notin \mathscr{O}$, then $\beta_1 < \beta_1^*$ and $\mathscr{W}^* > \beta_1 HW \geq \mathscr{W}$, contradicting the optimality of $\mathscr{W}^*$. Thus $S \in \mathscr{O}$.

COROLLARY. *If the dimensions of S are $H \times W$, then S is an optimal pattern and $\mathscr{W} = \mathscr{W}^*$.*

Proof. If S has dimensions $H \times W$, then $\mathscr{W} = \omega \leq \beta_1 HW$ since S was created by using the value β_1 in Algorithm One. By applying Theorem 2, we obtain the desired result.

To formulate corresponding results for Algorithm Two, we introduce definitions that are similar to those used for Algorithm One. Let $\beta_2{}^*$ be the smallest value of β_2 in Algorithm Two that allows the creation of some rectangle $S_2{}^*$ of $\mathcal{O}$. Recall that $\mathcal{O}$ is the set of guillotine rectangles with optimal waste $\mathcal{W}^*$. Define $\omega_2{}^*$ as the trim (inner) waste of $S_2{}^*$, and let $S_1, S_2, \cdots, S_t$ denote the sequence of partial rectangles that the algorithm builds into $S_2{}^*$. If $\omega_1, \omega_2, \cdots, \omega_t$ are the corresponding trim wastes of the S_i, then $\beta_2{}^* = \max_{1 \le i \le t}\{\omega_i/a(S_i)\}$ and $\omega_2{}^* = \omega_t$.

For example, the guillotine rectangle of Figure 5 would be built by the sequence of partial rectangles pictured in Figure 6, and each $\omega_i \le \beta_2{}^* a(S_i)$. Also, $S_4 = S_2{}^*$.

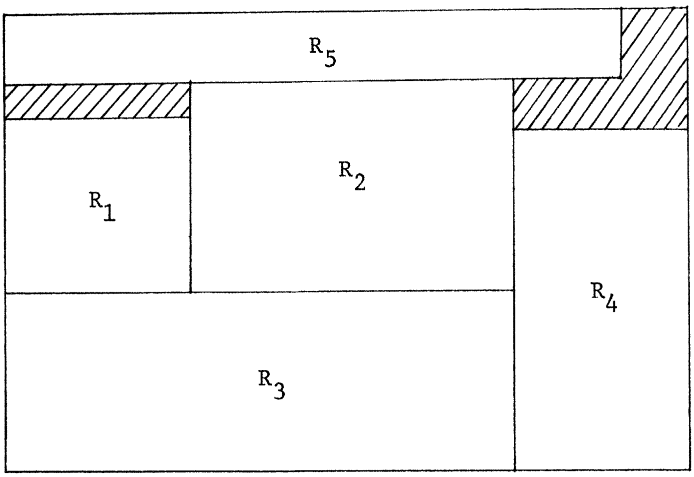

Figure 5. An optimal guillotine rectangle $S_2{}^*$.

If we define S_p to be the partial rectangle for which $\beta_2{}^* = \omega_p/a(S_p)$, then $\mathcal{W}^* \ge \omega_2{}^* \ge \omega_p = \beta_2{}^* a(S_p)$. Assuming that $S_2{}^*$ is not one of the R_1, $R_2, \cdots, R_n$, then $a(S_p) \ge a(\rho)$ where ρ is the guillotine rectangle having the smallest area of all rectangles constructed by one horizontal or vertical build of some R_i and R_j subject to the bound constraints. Thus, $\mathcal{W}^* \ge \beta_2{}^* a(\rho)$.

THEOREM 3. *If S is the guillotine rectangle of $L^{(M)}$ selected in Algorithm Two as the pattern with minimum total waste $\mathcal{W}$, and the algorithm was applied with a fixed value of β_2, then*

$$\mathcal{W} - \mathcal{W}^* \le |\mathcal{W} - \beta_2 a(\rho)|.$$

Proof. We consider two cases.

(i) Suppose $S_2{}^*$ is not one of the R_i. If $\beta_2 < \beta_2{}^*$, then $\mathcal{W} > \mathcal{W}^* \ge \beta_2 a(\rho)$ and $0 < \mathcal{W} - \mathcal{W}^* < \mathcal{W} - \beta_2 a(\rho)$. If $\beta_2{}^* > \beta_2$, then $\mathcal{W} - \mathcal{W}^* = 0 \le |\mathcal{W} - \beta_2 a(\rho)|$.

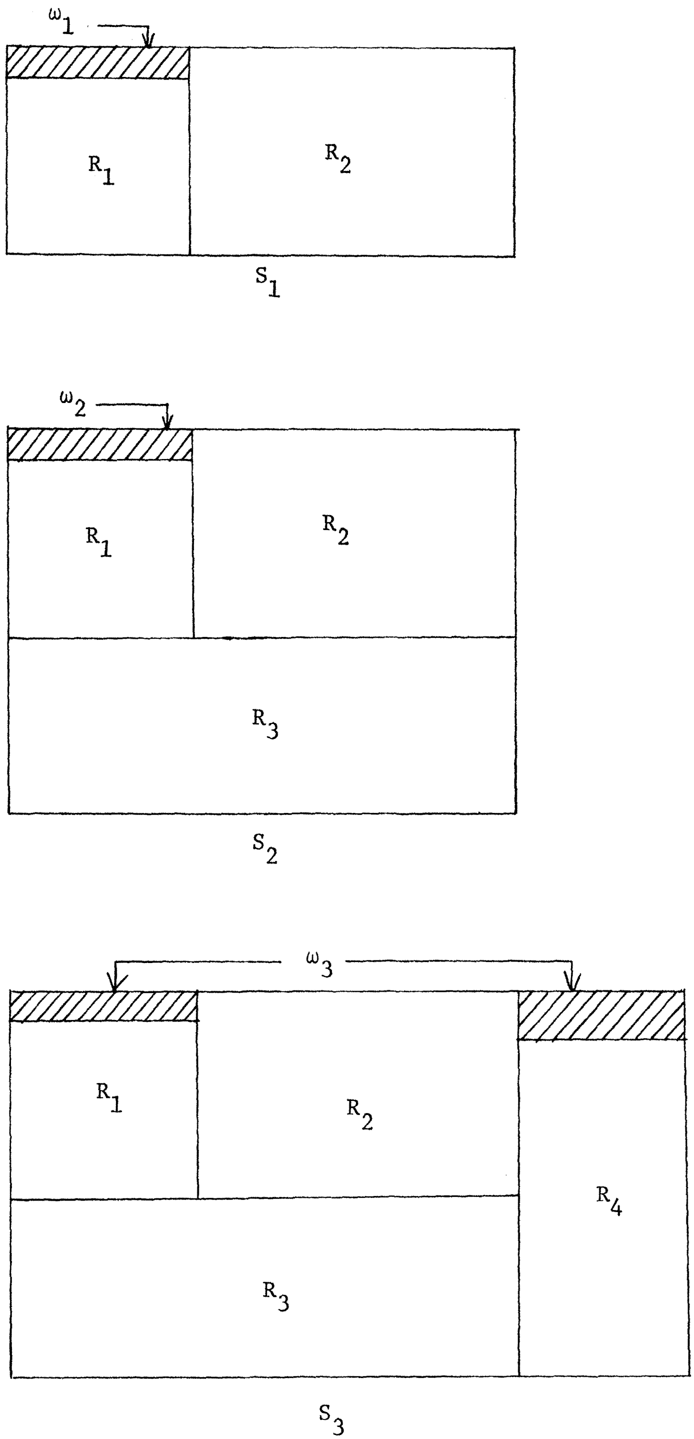

Figure 6. The partial rectangles which build into $S_2{}^*$.

(ii) If S_2^* is some R_i, then $\beta_2^* = 0$. Any value of β_2 satisfies $\beta_2 \geq \beta_2^*$ and $\mathscr{W} - \mathscr{W}^* = 0$.

THEOREM 4. *If the waste $\mathscr{W}$ of the pattern S obtained from Algorithm Two with a fixed value of β_2 satisfies $\mathscr{W} \leq \beta_1 a(\rho)$, then S is an optimal pattern and $\mathscr{W} = \mathscr{W}^*$.*

Proof. Assume $\mathscr{W} \leq \beta_2 a(\rho)$. If $S \notin \mathcal{O}$, then $\beta_2 < \beta_2^*$ and $\mathscr{W}^* > \beta_2 a(\rho) \geq \mathscr{W}$ contradicting the optimality of $\mathscr{W}^*$. Therefore, $S \in \mathcal{O}$.

This theorem relies on the assumption that the optimal guillotine rectangle is not one of the rectangles R_i. This is usually the case in the industrial problems we are interested in solving. Furthermore, the error bound of Theorem 3 is more effective if the area of ρ is close to the area of S_p. We note that there is no result for Algorithm Two that corresponds to the Corollary associated with Theorem 2.

In the discussion which follows, we suggest the values of β_1 and β_2 in Algorithms One and Two.

4. COMPUTATIONAL RESULTS

To effectively use the algorithms of Section 2, we recommend that the parameters β_1 and β_2 be selected with the following result in mind.

THEOREM 5. *$\beta_1^* \leq \mathscr{W}/(HW)$ and $\beta_2^* \leq \mathscr{W}/a(\rho)$ where $\mathscr{W}$ is the total waste of any guillotine pattern that cuts no more than b_i of any rectangle R_i from the stock sheet.*

Proof. Since $\mathscr{W}^* \leq \mathscr{W}$, $\beta_1^* HW \leq \mathscr{W}$ and $\beta_2^* a(\rho) \leq \mathscr{W}$.

For each of the algorithms, it is desirable to choose the parameter β_i as close as possible to, but greater than, its optimal value. This not only guarantees that the specified method finds an optimal solution, but also reduces the amount of computing time needed to generate the solution. From Theorem 5, we conclude that the waste $\mathscr{W}$ of any hand-generated pattern can serve as an upper bound on either β_1^* or β_2^*. This is particularly useful since people who deal with the constrained problem in industry have considerable expertise in creating good pattern layouts. The better the hand-generated solution, the smaller the bounds on β_1^* and β_2^*, and the faster an optimal solution can be found.

If Algorithms One and Two are executed with the same parameter values, i.e. $\beta_1 = \beta_2$, then the final set $L^{(M_2)}$ of rectangles generated in Algorithm Two is a subset of the guillotine rectangles $L^{(M_1)}$ generated by Algorithm One. As general procedure, it is recommended that Algorithm Two be applied first with β_2 set at approximately 0.05 or 5%. The waste $\mathscr{W}$ of the resulting best pattern provides an upper bound on β_1^*. In the cases where an optimal solution is desired, the optimality conditions associated with Algorithm One provide a better measure of the optimality of the best solution than do those of Algorithm Two, although the latter method usually finds an optimal solution faster.

The algorithms outlined in Section 2 also assumed that the rectangles R_i were of fixed orientation. This assumption is not strictly necessary, and Algorithms One and Two can be modified to allow for rotations of each rectangle of $L^{(k)}$ when constructing the horizontal and vertical builds. With $\mathscr{W}^*$ defined as the minimum waste attainable by any guillotine pattern containing possible rotations of the rectangles, the previous discussion of Section 3 concerning error bounds and optimality conditions remains valid.

TABLE I

An Example of a Constrained Cutting Stock Problem

i	R_i	b_i
1	17×9	1
2	11×19	4
3	12×21	3
4	14×23	4
5	24×15	1
6	24×15	2
7	25×16	4
8	27×17	2
9	18×29	3
10	21×31	3
11	32×22	2
12	23×33	3
13	34×24	2
14	35×25	2
15	36×26	1
16	37×27	1
17	38×28	1
18	39×29	1
19	41×30	1
20	43×31	1

To illustrate the use of these algorithms, we consider the following problem. It is a variation of an example presented by Christofides and Whitlock. Let $H \times W = 70 \times 40$, and assume the rectangles R_i are given in Table I. Tables II and III summarize the computational results obtained by applying Algorithms One and Two. By applying Theorem 2, we determine that Algorithm One finds the optimal solution when $\beta_1 = 0.03$. In contrast, Algorithm Two finds this solution with less computing time for $\beta_2 = 0.05$, but the optimality condition of Theorem 4 is not satisfied.

These results were obtained by coding the algorithms in FORTRAN and executing the programs on the UNIVAC 1100/81. The algorithms must save information concerning the rectangles, T, which are con-

TABLE II

COMPUTATIONAL RESULTS OBTAINED BY ALGORITHM ONE

β_1	0.01	0.02	0.03
Size of $L^{(M)}$	100	217	441
Best pattern S:			
Dimensions	69 × 33	70 × 39	70 × 40
Inner waste	0	36	79
Total waste	523	106	79
Theorem 1 (error bound)	495	50	5
Actual $\mathscr{W} - \mathscr{W}^*$	444	27	0
Theorem 2 (optimality?)	No	No	Yes
Computing time (sec)	23	34	73

$\beta_1^* = 79/2800 \simeq 0.0283$.

structed in Step 2.a. They can employ arrays to store the height and width of each T, as well as its composition, since the latter is required for Step 2.a(iii). In some instances, the amount of storage required for this task could be reduced. For example, if $n = 10$ and all bound constraints satisfy $b_i \leq 9$, then a single integer having 10 digits could be used to represent the composition of T. Each digit would represent the number of times a rectangle R_i appears in T.

Furthermore, Step 2.b requires the removal of duplicate patterns from $L^{(k)}$. To reduce the amount of searching needed, we could form each new set $F^{(k)}$ by building the rectangles of $F^{(k-1)}$ together with the rectangles of $L^{(k-1)}$. In addition, one may initially order the rectangles, R_i, by some criterion such as increasing heights, in order to minimize the possibility of generating a pattern that has already been created.

5. THE GENERAL CUTTING STOCK PROBLEM

The algorithms we have presented may also be used to obtain approximate solutions to a general stock cutting problem which can be stated as follows. Let $R_1, R_2, \cdots, R_n$ be a set of rectangles where each R_i has height h_i and width w_i. Define $d_1, d_2, \cdots, d_n$ to be the number of each

TABLE III

COMPUTATIONAL RESULTS OBTAINED BY ALGORITHM TWO

β_2	0.01	0.015	0.02	0.03	0.04	0.05	0.10
Size of $L^{(M)}$	54	74	104	152	238	404	2673
Best pattern S:							
Dimensions	69 × 33	70 × 39	70 × 39	70 × 39	70 × 40	70 × 40	70 × 40
Inner waste	0	36	36	36	100	79	79
Total waste	523	106	106	106	100	79	79
Theorem 3 (error bound)	518.7	99.7	97.6	93.5	83.3	59.1	37.2
Actual $\mathscr{W} - \mathscr{W}^*$	444	27	27	27	21	0	0
Theorem 4 (optimality?)	No	No	No	No	No	No	No
Computing time	21″	22″	24″	27″	36″	64″	23′47″

$\beta_2^* = 36/840 \simeq 0.0429$, $a(\rho) = 418$.

respective rectangle that is ordered by a customer. If there are m standard stock sizes $H_1 \times W_1, H_2 \times W_2, \cdots, H_m \times W_m$ which can be used to fill the order, find the set of patterns that cuts the customer order with minimum total trim waste.

A solution to this general problem can be obtained in the following manner. Define a constraint matrix A to be the $n \times k$ matrix that consists of all k possible guillotine patterns that can be used to cut the rectangles R_i from the stock sheets. Each column a_p represents a pattern p, where a_{ip} is the number of times rectangle R_i appears in pattern p. If each pattern has trim waste c_p, then the problem can be formulated as

$$\text{minimize} \quad \textstyle\sum_{p=1}^{k} c_p x_p$$

$$\text{subject to} \quad A \cdot \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_k \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \cdot \\ \cdot \\ d_n \end{bmatrix}$$

and $x_p \geq 0$ and integer.

In most cases, it is impractical to solve this problem. However, an approximate solution can be found by replacing the equality constraint with a greater than or equal to condition, and by relaxing the integer requirements of the x_p. The size of the matrix A is also reduced by eliminating patterns that are not desirable. The modified problem can then be solved by linear programming methods, and the solution rounded to integer values.

Algorithms One and Two may be used to generate guillotine cutting patterns for the constraint matrix A. We can apply either of our algorithms with $H = \max_{1 \leq i \leq m}\{H_i\}$ and $W = \max_{1 \leq i \leq m}\{W_i\}$. The constraints b_i for the individual R_i may be defined as $b_i = \lfloor HW/(h_i w_i) \rfloor$ for $i = 1, 2, \cdots, n$. After executing Algorithm One or Two with these values, we select a subset of acceptable patterns from the final list $L^{(M)}$. For example, the criterion for choosing the patterns could be that their total trim waste is less than a fixed percentage of the area of one of the m stock sizes, if they are thought of as cutting patterns for that stock size.

We applied the technique described above to the problems listed in Tables IV and V, which are taken from the lumber industry. In the first problem, 10 rectangles are to be cut from three sizes of stock in the quantities specified in the table. By taking $H \times W = 66 \times 96$ and using the listed values for the constraints b_i, we applied Algorithm One with $\beta_1 = 0.01$ after enlarging all rectangle and stock dimensions by 0.187 inches (the width of the saw blade). Within the algorithm, we permitted rotations of the rectangles in the vertical and horizontal builds. Ninety

TABLE IV

CUTTING STOCK PROBLEM ONE

Rectangles				Stock Sheets		
i	R_i	d_i	b_i	j	$H_j \times W_j$	Quantity
1	$29\frac{7}{8} \times 41\frac{1}{8}$	200	5	1	48×96	Unlimited
2	$29\frac{7}{8} \times 35\frac{1}{8}$	300	6	2	66×96	Unlimited
3	$29\frac{7}{8} \times 32\frac{1}{8}$	200	6	3	34×96	163
4	$34 \times 23\frac{1}{8}$	350	8			
5	$29\frac{7}{8} \times 26\frac{1}{8}$	200	8			
6	$17\frac{7}{8} \times 41\frac{1}{8}$	150	8			
7	$29\frac{7}{8} \times 23\frac{1}{8}$	700	9			
8	$34 \times 17\frac{1}{8}$	400	9			
9	$14\frac{7}{8} \times 32\frac{1}{8}$	200	9			
10	$11\frac{7}{8} \times 35\frac{1}{8}$	200	9			

guillotine rectangles were selected from the final list $L^{(M)}$. We treated each of these rectangles as a pattern that could cut one of the three stock sheets, thereby producing a trim waste that was less than 5% of the area of that stock sheet. Solving the linear programming formulation of the problem with these 90 patterns yielded a solution consisting of 10 of these patterns. Rounding the solution to integers, we obtained an approximate solution that required 2,311,488 square inches of stock material, 5.3% of which was considered waste. The results compare favorably with the company provided solution which required 2,495,040 square inches of cutting stock, 12.3% of which was waste. Our approach required a computing time on the UNIVAC 1100/81 of 19 minutes for Algorithm One and 20 seconds for the linear programming problem.

The second problem we considered is taken from Skalbeck and Schultz. Table IV specified the size and ordered quantities of the rectangles R_i and the available stock sizes. Setting $H \times W = 60 \times 108$ and all $b_i = 9$, we applied Algorithm One with $\beta_1 = 72/6480$ and terminated it when the size of $L^{(k)}$ reached a limit of 5000 rectangles. We again permitted rotations of the rectangles in the algorithm. From the 5000 rectangles, the algorithm selected 97 acceptable patterns. The trim wastes of these rectangles, when considered as patterns cut from one of the stock sheets,

TABLE V

CUTTING STOCK PROBLEM TWO

i	R_i	d_i	b_i	j	$H_j \times W_j$	Quantity
1	28×30	180	9	1	48×96	Unlimited
2	20×24	180	9	2	60×108	Unlimited
3	16×20	100	9			
4	14×21	100	9			
5	12×18	100	9			

was less than 3% of the area of that stock sheet. The rounded solution to the linear programming problem required 326,736 square inches of stock, 1.9% of which was waste. The solution obtained by Skalbeck and Schultz required 331,344 square inches of stock, 3.2% of which was waste. The building algorithm required was 12 minutes, 38 seconds of computing time on the UNIVAC 1100/81, while solving the linear program required 20 seconds.

6. CONCLUDING REMARKS

As discussed in the previous section, the methods described in this paper have successfully obtained approximate solutions to rectangular cutting stock problems. We also note that for a given problem, the set of patterns $L^{(M)}$ generated by our algorithms could be reused to fill other customer orders for sets of the same rectangles. This is easily accomplished by selecting different patterns from $L^{(M)}$ and changing the coefficient matrix of the linear programming problem.

Furthermore, our algorithms are fast and efficient solution methods when applied to constrained cutting problems for which n and b_i are not large. For large constrained problems, the approximate solutions that are calculated by our algorithms are governed by the error bounds proved in Section 3, thus giving our methods an additional advantage over other approximation techniques.

REFERENCES

BROWN, A. R. 1971. *Optimum Packing and Depletion: The Computer in Space- and Resource-Usage Problems.* American Elsevier, New York.

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

DYSON, R. G., AND A. S. GREGORY. 1974. The Cutting Stock Problem in the Flat Glass Industry. *Opnl. Res. Quart.* **25,** 41–53.

SKALBECK, B. A., AND H. K. SCHULTZ. 1976. Reducing Trim Waste in Panel Cutting Using Integer and Linear Programming. *Proc. Western AIDS Conf.* (March).