Reprinted from OPERATIONS RESEARCH
Vol. 13, No. 1, January-February, 1965
Printed in U.S.A.

MULTISTAGE CUTTING STOCK PROBLEMS OF TWO AND MORE DIMENSIONS

P. C. Gilmore and R. E. Gomory†

Thomas J. Watson Research Center, Yorktown Heights, New York

(Received January 22, 1964)

In earlier papers [*Opns. Res.* 9, 849–859 (1961), and 11, 863–888 (1963)] the one-dimensional cutting stock problem was discussed as a linear programming problem. There it was shown how the difficulty of the enormous number of columns occurring in the linear programming formulation could be overcome by solving a knapsack problem at every pivot step. In this paper higher dimensional cutting stock problems are discussed as linear programming problems. The corresponding difficulty of the number of columns cannot in general be overcome for there is no efficient method for solving the generalized knapsack problem of the higher dimensional problem. However a wide class of cutting stock problems of industry have restrictions that permit their generalized knapsack problem to be efficiently solved. All of the cutting stock problems that yield to this treatment are ones in which the cutting is done in stages. In treating these practical cutting problems, one often encounters additional conditions that affect the solution. An example of this occurs in the cutting of corrugated boxes, which involves an auxiliary sequencing problem. This problem is discussed in some detail, and a solution described for the sequencing problem under given simplifying assumptions.

IN EARLIER papers,[1,2] we have discussed the one-dimensional stock cutting or trim problem as a linear programming problem. In this paper we take up the corresponding problem in higher dimensions. For clarity, we will state here a version of what we call the two-dimensional problem:

Consider a supply of stock rectangles of width W and length L, and a demand for N_i rectangles of width w_i and length l_i, $i=1, \cdots, m$. A two-dimensional problem is to cut the stock rectangles into the smaller demand rectangles using as few stock rectangles as possible.

Problems resembling the one just described turn out to be surprisingly common in applications. Specific examples were given by KANTOROVITCH[3] in his very early discussion of the trim problem, and by REITH.[4] The reason why these problems are common seems to be quite fundamental. Just as economics of scale lead to some products being produced in large lots, they also tend to make economical the production or provision of some materials in large size. These economical sizes must then be cut up to

† This research was supported in part by the Office of Naval Research under Contract No. Nonr 3775(00), NR 047040.

meet consumer demands. Among obvious examples of this are the production of glass, metal sheets, graphite, photographic film, plastic, etc. Another, less obvious but illuminating, example is railway freight transportation, which is provided in boxcar sizes and which one can regard as being cut up for use into smaller crate sizes when crates are packed into the boxcar.

A second aspect of the higher dimensional problem in the very general form just described is its intractability. The example described above, when the small rectangles are to be fitted into the stock rectangles in any way, is basically of the jigsaw puzzle variety and methods are not now known for large-scale problems of this sort.

However, an examination of industrial problems shows that the problem stated above is unnecessarily general. In a great many industrial situations, a piece of material being cut must be cut all the way through from one edge to another. The resulting smaller pieces may then be treated separately and cut again, but again each is cut all the way through. The cutting patterns produced in this way will still be intricate and tremendously numerous, but, as we will see, this replacement of the original multidimensional problem with what might be called a multistage problem does lead to methods of solution.

In the first section we will review briefly the one-dimensional cutting stock problem, as described in reference 1, showing how a column generation technique for the problem requires a solution to a generalized knapsack problem. We will also describe a slightly different method for recursively solving the knapsack problem. We then show how a two-dimensional cutting stock problem can be posed as a linear programming problem and how column generating techniques now lead to more general knapsack problems for which no methods of solution are known. Some special kinds of two-dimensional problems are solvable, however, and one of these is discussed at length in the third section. Next we discuss other two-dimensional problems and a solvable three-dimensional problem. In the fifth section we take up one special two-dimensional problem, the corrugated box problem, and then discuss a related sequencing problem, the solution of which can affect the original cutting problem, as we show in the final section.

ONE-DIMENSIONAL CUTTING STOCK PROBLEM

The Linear Programming Formulation

By way of introduction to the present paper we will review some of our earlier work (references 1 and 2).

A one-dimensional noninteger cutting stock problem is the following: a continuous sheet of stock material of breadth L is to be cut so as to satisfy

demands for amounts N_i (lengths) of strips of breadth l_i, $i=1, \cdots, m$. The demand N_i can be satisfied by supplying any number of pieces of the strips of breadth l_i so long as the total lengths sum to at least N_i. As illustrated in Fig. 1, the demands are met by deciding upon various slitting patterns for the sheet of breadth L; the jth slitting pattern is a way of dividing the breadth L into the smaller breadths l_i, $i=1, \cdots, m$, and is applied to an amount x_j of the sheet.

In a linear programming formulation of a one-dimensional noninteger cutting stock problem, the matrix A of the linear programming problem will have m rows and a large number of columns, one for each of the possible slitting patterns. Thus each vector $[a_1, a_2, \cdots, a_m]$ of nonnegative integers a_i satisfying

$$L \geqq l_1 a_1 + l_2 a_2 + \cdots + l_m a_m \tag{1}$$

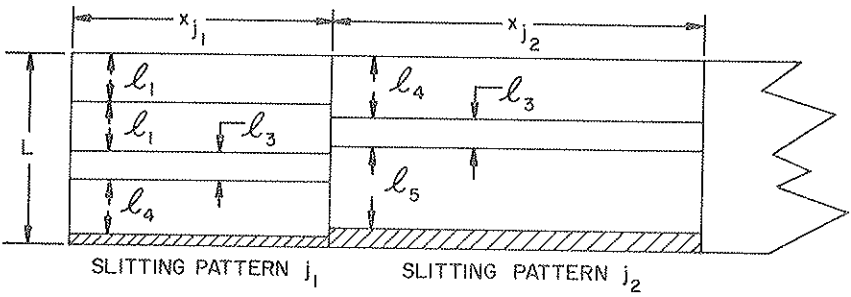

Figure 1

is a column of the matrix. If X is a column vector of variables, one for each column of A, and if 1 is a row vector of all 1's then the linear programming problem is stated:

$$\begin{cases} \text{minimize } 1 \cdot X, \\ \text{subject to } A \cdot X = N, \end{cases} \tag{2}$$

where N is the column vector $[N_1, N_2, \cdots, N_m]$ of demands N_i.

Here we have stated (2) without slack variables so that the possibility of overproducing one of the demanded breadths l_i is not admitted. In reference 1 we showed that the more general problem in which a slack variable is introduced into each equation does not lead to a cheaper solution. We have therefore omitted the slack variables here for the sake of simplicity and refer the reader to reference 1 for a discussion of the problem with slack variables.

Practical problems occur in both integer and noninteger forms, where by an integer problem we mean one in which the demands N_i are in integers and the variables x_j are restricted to being integer. Although our linear

programming methods will only solve the noninteger problem, practice has shown that rounded answers to the noninteger problem can often provide satisfactory solutions to integer problems.

The Knapsack Problem

The only computational difficulty standing in the way of solving a cutting stock problem as a linear programming problem is generally the very large number of columns in the problem. However in reference 1 we showed how this computational difficulty in the one-dimensional cutting stock problem could be reduced to the solution of a knapsack problem at every simplex pivot step. The knapsack problem is the following: let $\Pi_1, \Pi_2, \cdots, \Pi_m$ be the Lagrange multipliers or shadow prices associated with the m equations in a basic feasible solution to the linear programming problem, then the problem is

$$\begin{cases} \text{maximize } \Pi_1 a_1 + \Pi_2 a_2 + \cdots + \Pi_m a_m, \\ \text{subject to (1) and subject to } a_i,\ i = 1, \cdots, m, \\ \text{being nonnegative integers.} \end{cases} \tag{3}$$

In references 1 and 2 we described two methods for solving the knapsack problem, one of which is a recursive calculation. We would like to describe here another recursive calculation that has several advantages over the previous one when no cutting knife limitation, of the type described in reference 2, occurs. Both the present recursive calculation and the one described in references 1 and 2 can be regarded as extensions of the recursion in DANTZIG[5] or BELLMAN[6] to a knapsack problem in which the variables may be any nonnegative integers, and not just 0 or 1.

If we define $F_s(x)$ as the value of the best combination that can be fitted into a stock of length x using only the first s variables (i.e., $y_i = 0$, $i > s$), we have the recursion

$$F_s(x) = \max\{\Pi_s + F_s(x - l_s), F_{s-1}(x)\}, \quad \text{for} \quad s > 1. \tag{4}$$

This follows from the fact that the pattern yielding the value $F_s(x)$, $s > 1$, either does or does not use the sth variable at a positive level. If it does use it, then $F_s(x) = \Pi_s + F_s(x - l_s)$, and if it does not then $F_s(x) = F_{s-1}(x)$. Since $F_1(x) = \Pi_1[x/l_1]$ is easily obtained, and $F_s(0) = 0$ for all s, (4) enables us to calculate $F_s(x)$ in terms of $F_{s-1}(x)$ and $F_s(x')$, for $x' < x$, i.e., in terms of functions already known. Then $F_m(L)$ gives us the value of the best knapsack pattern. It should be noted that when $F_m(L)$ has been computed, so has $F_s(x)$ for all s, $1 \leqq s \leqq m$, and all x, $0 \leqq x \leqq L$.

Although the above calculation produces the best value, there still remains the question of finding the pattern that gave that value. To do this, we backtrack the recursion (4). Backtracking is simplified by the fol-

lowing device. Associate with $F_1(x)$ the index 1 if $x \geqq l_1$ and 0 otherwise. As $F_s(x)$, $s>1$, is computed for each x, associate with $F_s(x)$ the index s, if $F_s(x) = \Pi_s + F_{s-1}(x-l_s)$, and otherwise the index associated with $F_{s-1}(x)$. Consequently, the index associated with $F_s(x)$, for any s and x, is the largest index r for which a_r is positive in the pattern yielding $F_s(x)$. Therefore, to find the pattern yielding $F_m(L)$, we look first at the associated index, if this is r and $r>0$, then a_r is in the pattern at a value of 1 or more. We look next at $F_m(L-l_r)$, if its associated index is r' ($r'=r$ is possible) and $r'>0$, then a piece of breadth $l_{r'}$ was used in $F_m(L-l_r)$ and hence in $F_m(L)$. Continuing in this way, we obtain the whole pattern.

Two remarks are now in order. First, in computing (4) one need retain

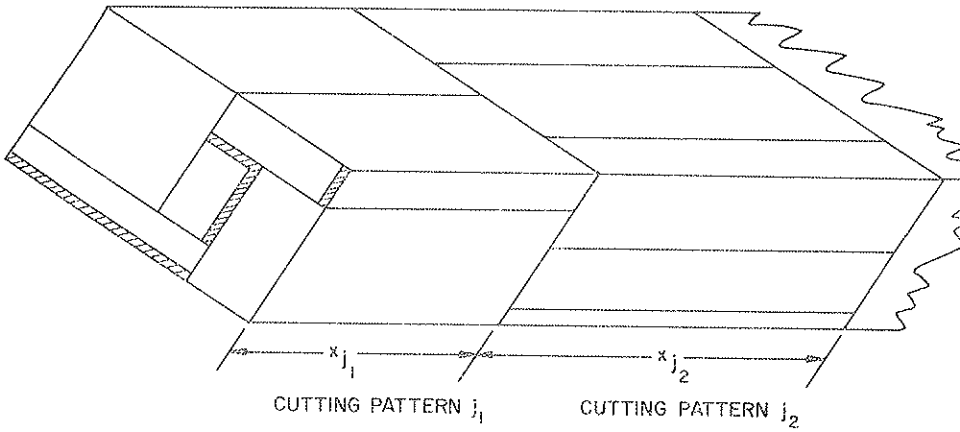

Figure 2

only $F_{s-1}(x)$ to compute $F_s(x)$, and in backtracking, one requires only $F_m(x)$; thus all values $F_{s-2}(x)$ can be discarded before beginning the computation of $F_s(x)$. Second, this recursion involves less arithmetic than the recursive method described in references 1 and 2 by a factor of approximately α where

$$\alpha = 1/m(\textstyle\sum_i [L/l_i]).$$

It may therefore be comparable in speed with the lexicographic method described under "Knapsack Method" in reference 2, although, in contrast to the lexicographical methods it does not seem to be adaptable to problems where the number of cutting knives is limited.

GENERAL TWO-DIMENSIONAL PROBLEM

The problem we are aiming at is the one described in the introduction, the problem of producing small rectangles by cutting up large ones. How-

ever, just as the roll cutting problem is approached by solving the related noninteger problem of slitting a sheet, this rectangle problem is approached by the noninteger problem of slitting a solid rectangular bar.

In this problem the stock material consists of a continuous bar of rectangular cross section $W \times L$, which is to be cut to supply an amount N_i, measured along the bar, of a bar of rectangular cross section $w_i \times l_i$, $i = 1, \cdots, m$. The amount N_i, for any i, may be supplied in any number of pieces. The cutting is performed as follows: a number of rectangular cutting patterns are decided upon, each pattern being described by a way of fitting the small rectangles $w_i \times l_i$ into the large one $W \times L$; the jth such pattern describes how the stock bar is to be slit a distance x_j to produce smaller bars, as illustrated in Fig. 2. The problem then is to supply the ordered amounts by cutting as little stock as possible.

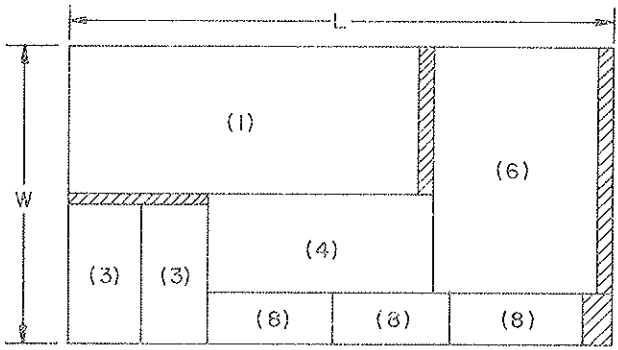

Figure 3

The linear programming problem that is equivalent to such a cutting stock problem has exactly the form (2). However the columns of A are now defined as follows. To each possible rectangular cutting pattern of $W \times L$ there will correspond a column $[a_1, a_2, \cdots, a_m]$ of A, where a_i is the number of rectangles $w_i \times l_i$ which occur in the pattern. For example in Fig. 3, a rectangular cutting pattern is illustrated in which 1 rectangle $w_1 \times l_1$ occurs, 2 rectangles $w_3 \times l_3$, 1 rectangle $w_4 \times l_4$ and 3 rectangles $w_8 \times l_8$. Hence if $m = 10$ for a problem with this as a possible cutting pattern, then a column in the matrix of the corresponding linear programming problem is [1, 0, 2, 1, 0, 0, 0, 3, 0, 0].

As in the one-dimensional case, the difficulty in solving this cutting stock problem as a linear programming problem is the immense number of columns that can occur in the matrix. If, as in the one-dimensional case, we apply a column generation technique then the following generalized knapsack problem arises: given that Π_i is the Lagrange multiplier or

shadow price corresponding to the ith equation, $i=1, \cdots, m$, in some basic feasible solution then

$$\begin{cases} \text{maximize } \{\Pi_1 a_1+\Pi_2 a_2+\cdots+\Pi_m a_m\}, \\ \text{subject to the condition that } [a_1, a_2, \cdots, a_m] \\ \text{correspond to a cutting pattern fitting a rectangle} \\ w_i \times l_i,\ i=1, \cdots, m, \text{ into a rectangle } W \times L. \end{cases} \tag{5}$$

We know of no economical method for solving this generalized knapsack problem.

Fortunately, however, there appear to be many practical cases where it is not necessary to solve the completely general two-dimensional problem

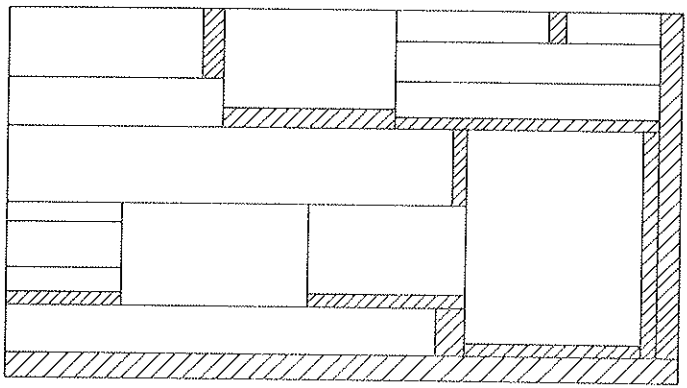

Figure 4

Clearly, since economy has led to the production of large sizes, inexpensive cutting methods can also be expected. Inexpensive cutting methods appear frequently to have one common characteristic: a cut in a piece of material must begin on one side of the material and traverse the material in a straight line to the other side. This is the kind of cut made by many types of machinery and used in many industries. One example is the guillotine cutter used in cutting paper sheets, and for this reason we call this type of cut a guillotine cut.

Restricting the permissible cuts in a two-dimensional cutting pattern to guillotine cuts severely limits the permissible patterns. For example, the pattern of Fig. 3 cannot be produced by guillotine cuts. But quite general patterns are still possible as illustrated in Fig. 4. A recursion can be written for the general guillotine case‡ but in this paper we confine

‡ The recursion is

$$G(x,y)=\max_{\substack{0\leqq x_0<x/2 \\ 0\leqq y_0<y/2}} \{g(x,y),\ G(x_0,y)+G(x-x_0,y),\ G(x,y_0)+G(x,y-y_0)\}$$

ourselves to the computationally easier special subclasses of patterns that correspond to the cutting methods used in certain industries.

Before discussing some of the special subclasses of patterns, we wish to dispose of an apparent complication that can arise because some materials are isotropic while others are not. For isotropic materials such as glass the orientation of an ordered rectangle $w_i \times l_i$ with respect to the stock rectangle $W \times L$ is irrelevant while with nonisotropic materials, such as corrugated paper, the orientation can be relevant. Actually this leads to no difficulties. As far as the generalized knapsack problem is concerned we will always assume that an ordered rectangle $w_i \times l_i$ must be oriented in any pattern in the same way as the stock rectangle $W \times L$, and if indeed the orientation of a rectangle $w_i \times l_i$ is irrelevant in a cutting stock problem, then we will also assume that a rectangle of width l_i and length w_i is among the ordered rectangles and has the same price. In the linear programming problem we will always have exactly one equation for each demanded rectangle no matter what orientations of it are permitted, and the entry a_i in any column will be the number of rectangles $w_i \times l_i$ occurring in the corresponding cutting pattern counting both orientations.

TWO-STAGE GUILLOTINE CUTTING

Usual Formulation

An important subclass of patterns involving only guillotine cuts are those which can be thought of as taking place in two stages. Figure 5 illustrates a pattern produced in this way: first the rectangle $W \times L$ is slit down its length into strips and then each of these strips is taken individually and chopped across its width. Sometimes a third trimming stage is permitted as illustrated in Fig. 6, where the rectangles produced by the first two stages of cutting are trimmed down their length to produce a rectangle of an ordered size. Generally the knapsack problem arising when trimming is permitted (we call this the nonexact case) is more difficult than the one arising when trimming is not permitted (the exact case); therefore, unless otherwise stated, we assume that trimming is permitted.

In order to solve the two-stage guillotine cutting problem it is sufficient to solve the corresponding generalized knapsack problem

$$\text{maximize } \Pi_1 a_1 + \Pi_2 a_2 + \cdots + \Pi_m a_m \text{ subject to the condition that } [a_1, a_2, \cdots, a_m] \text{ correspond to a two-stage guillotine cutting pattern.} \tag{6}$$

---

where $G(x,y)$ is the maximum value that can be obtained from an $x \times y$ rectangle using $w_i \times l_i$ rectangles at price Π_i and any succession of guillotine cuts, and $g(x,y) = \max_{\left[\substack{w_i \leqq x \\ l_i \leqq y}\right]} \Pi_i$.

This knapsack problem can be solved in two stages:

(i) for all widths w_i calculate Π_i^*, the optimum value obtainable by fitting rectangles $w_j \times l_j$, where $w_j \leqq w_i$, end to end into a strip of width w_i and length L. For each i this is a knapsack problem of the type (3).

(ii) The optimum value of the objective function $\Pi_1 a_1 + \cdots + \Pi_m a_m$ is then obtained by solving one more knapsack problem: max $\Pi_1^* b_1 + \cdots + \Pi_m^* b_m$ subject to $W \geqq w_1 b_1 + \cdots + w_m b_m$.

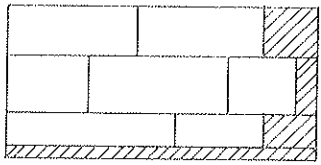

Figure 5

The knapsack problems of (i) above can all be solved together by one recursive calculation of the kind described in the section, "The Knapsack Problem," the first section of this paper. For let the demanded rectangles be reindexed so that $w_1 \leqq w_2 \leqq \cdots \leqq w_m$. Then $\Pi_i^* = F_i(L)$ and therefore

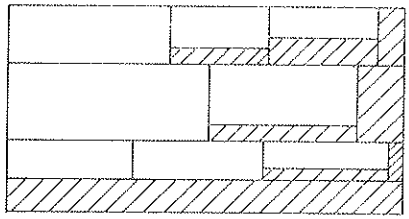

Figure 6

as we pointed out above, the Π_i^* for all i are calculated in the course of calculating $F_m(L)$. Hence the two-stage guillotine cutting knapsack problem can be solved by solving two knapsack problems of the type (3).

Formulation as a Staged Linear Programming Problem

It is also possible to write the two-stage guillotine problem in another way as a linear programming problem and apply a mixture of the approach of reference 1 and the decomposition method of Dantzig and Wolfe.[7] In this second approach, the problem is treated as a two-stage linear programming problem with the first stage corresponding to the process of slitting the stock rectangles into strips with widths corresponding to the widths of demanded rectangles, and with the second stage corresponding to the process of chopping the strips into demanded lengths.

Let A now be a matrix of $2m$ rows partitioned vertically into $m+2$ matrices A_s. Let $\bar{x}$ be a column vector partitioned horizontally, to conform with A, into $\bar{x}_0, \bar{x}_1, \cdots, \bar{x}_m, \bar{u}$, where $\bar{u}$ are slack variables. Finally let $\bar{N}$ be a $2m$-vector partitioned into a zero m-vector and the m-vector $\bar{N}$ of demands for rectangles, ordered as before in increasing size of the widths of demanded rectangles.

The columns of A_0 correspond to the activities that slit the stock rectangles into strips. Specifically, in the jth column, which corresponds to the jth strip slitting pattern, the first m elements are nonnegative integers $b_1 \cdots b_m$ satisfying $\sum_i b_i w_i \leqq W$, and the last m elements are all 0. There is one such column of A_0 for every set of integers $b_1, \cdots, b_m$ satisfying this inequality, corresponding to the activity of slitting the stock rectangle into b_i strips of width w_i, for $i=1, \cdots, m$.

A_s, for $s=1, \cdots, m$, can be regarded as the set of activities that take strips of width w_s and chop them into rectangles. The columns of A_s then contain zeros in the top m rows except for -1 in the sth row. There are nonnegative integers a_i in the rows $m+i$, $i=1, \cdots, s$, and zeros below that, the integers satisfying the inequality $\sum_i a_i l_i \leqq L$. For each set of integers satisfying this inequality there is a column of A_s corresponding to the activity of chopping a strip of width w_s into a_i pieces of length a_i, $i=1, \cdots, s$.

Finally A_{m+1} is a $2m \times m$ matrix with the first m rows all zeros and the last m rows $-I$, it being a matrix of coefficients for slack variables for the last m equations. The problem can then be stated:

$$\begin{cases} \text{Minimize } \sum_j x_j^0, \\ \text{subject to } A \cdot \bar{x} \geqq \bar{N}. \end{cases} \tag{7}$$

$x_1^0, x_2^0, \cdots$ are the numbers of times each of the various slitting patterns for forming strips are used and $x_1^s, x_2^s, \cdots$ are the numbers of times each of the various patterns for chopping the strips of width w_s are used.

By defining A_0^* to be the first m rows of A_0, and A_s^* for $s=1, \cdots, m$ to be the rows $m+1, \cdots, m+s$ of A_s, the equations of (7) have the following form:

$$\begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
 & -1, \cdots, -1 & 0 & & 0 & & \bar{x}_0 & & 0 \\
A_0^* & 0 & -1, \cdots, -1 & & 0 & 0 & \vdots & & \vdots \\
 & & 0 & & -1, \cdots, -1 & & \bar{x}_1 & & \\
\hline
 & A_1^* & & & & & \vdots & = & N \\
 & & A_2^* & & & & & & \\
\hline
0 & 0 & 0 & & A_m^* & -I & \bar{x}_m & & \\
 & & & & & & \bar{u} & & 0 \\
\hline
\end{array}$$

The first m equations state that exactly the strips produced by the first cutting are used in the second stage. The second set of m equations state that the demands for rectangles must be met by cutting up the strips.

One can then either treat problem (7) directly in the style of reference 1 or, alternatively, apply the method of decomposition of Dantzig and Wolfe.[7]

In the direct approach, the prices of a starting solution would be applied to find the most profitable column of A_0. This would be one knapsack calculation. To find the most profitable column among all the A_s, $m+1 \geqq s \geqq 1$, would be a series of calculations, all of which can be subsumed in one recursive knapsack calculation of the type described above; the slack variables can be treated in the usual way. The work of finding an improving column seems therefore to be about the same in both cases, but the inverse in this second approach is (usually) $2m \times 2m$ instead of $m \times m$, and, in addition, if one takes past experience as being applicable here, we should expect twice as many pivot steps in the second approach. These differences become more pronounced in higher dimensions.

A smaller inverse can be obtained by a Dantzig-Wolfe decomposition that splits the matrix A into an upper part A_1, consisting of the first m equations, and a lower part A_2 consisting of the last m. The inverse with which one works would then become $m \times m$. However, for each pivot step on the problem with matrix A_1 a complete linear programming trim problem for the matrix A_2 has to be solved, since the problem must be solved of chopping strips of certain widths, and costing certain amounts, into the desired rectangles at minimum cost.

OTHER TWO- AND THREE-DIMENSIONAL PROBLEMS

In the preceding section we saw that the generalized knapsack problem (5) is solvable when the cutting patterns are producible by two-stage guillotine cutting. Next we discuss several other two-dimensional cutting patterns and then go on to discuss the problem of many stock rectangles for two-stage guillotine cutting. We will then discuss a three-dimensional cutting stock problem. Finally we will return to two-dimensional problems in which the stock may have defects making portions of it unusable.

For some of these problems we will describe solutions, while others remain unsolved.

Some Two-Dimensional Cutting Problems

We now describe some special subclasses of the two-stage guillotine cutting patterns that have been suggested to us by special applications in the glass and steel industries. These subclasses are ones in which the second stage cutting must be performed simultaneously on some of the

strips resulting from the first stage cutting. So, for example, it may be necessary to divide the strips into two groups and cut together all the strips in one of the groups. Figure 10 illustrates a pattern that results when exactly 2 groups must occur, while Fig. 7 illustrates a pattern that results when exactly 1 group must occur. In general there will be exactly p groups of strips for some determined p. The two-stage problem discussed previously, however, can be regarded as a case when p is unlimited; for this reason we will refer to this as a free two-stage problem.

(1) The 1-Group Problem

The class of completely grouped patterns, that is when $p=1$, is much more restricted than the free two-stage class. In spite of this, the corresponding knapsack problem in the nonexact case appears to be much more difficult than the one for the free two-dimensional case. A variant of the

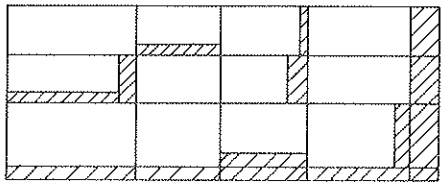

Figure 7

lexicographical scheme described in reference 2 looks possible but has not been explored.

Whether or not the cutting is exact greatly affects the difficulty of problems of this type. In a nonexact problem, the demand for a rectangle $w_i \times l_i$ can be filled by any $w \times l$ rectangle whose dimensions satisfy $w \geqq w_i$ and $l \geqq l_i$. For example, in the nonexact case, the strip shown in Fig. 8 would be considered as supplying three rectangles of dimensions $w_1 \times l_1, w_2 \times l_2$, and $w_3 \times l_3$. In the exact case, it would supply only a rectangle of dimensions $w_3 \times l_3$.

In contrast to the nonexact the knapsack problems for the exact case often become trivial. For if we can assume, $l_i \neq l_j$ if $i \neq j$, then only one width (and only the corresponding length) can be used in any pattern.

In fact, if we only assume that $w_i \neq w_j$ implies $l_i \neq l_j$, the problem remains easy. This relaxation allows several different lengths to be demanded with the same width and permits patterns such as the one illustrated in Fig. 9. To solve the knapsack problem for this case, one evaluates the value v_i of a strip of width w_i by any suitable method using only the associated l's. The total value for the rectangle is then $\max_i v_i[W/w_i]$. In general there are at most m ordinary knapsack problems of the kind (3) to be solved,

one for each w_i; however, the total number of variables appearing in these knapsack problems does not exceed m.

(2) The 2-Group Problem

This problem is encountered in the glass industry. Being a combination of two 1-group problems, this seems to be even more difficult than (1) in the nonexact case. Again in the exact case the assumption, that for any i and j if $(w_i \neq w_j)$ then $(l_i \neq l_j)$, gives a tractable problem. Once the v_i associated with each strip of width w_i is found for all i, one does a knapsack problem on the width dimension, in which only two distinct w_i are allowed. This results automatically in the patterns of Fig. 10.

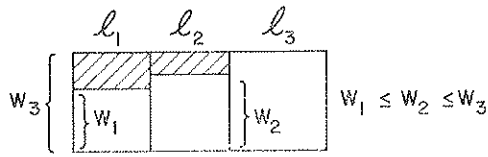

Figure 8

(3) Return to the Same Dimension, or 3-Stage Guillotine Cutting

In a third stage of guillotine cutting, one returns to the width dimension once cut to cut it again. This was done before in the trimming process on the nonexact problems, but it did not fundamentally affect the methods

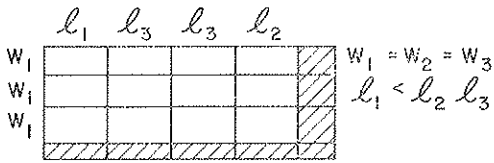

Figure 9

of solution. With general third-stage cutting permitted the previous methods of solution are no longer applicable. Problems of this sort are encountered in both the cutting of paper sheets and of glass. An example of a pattern cut by 3-stage guillotine cutting is given in Fig. 11; that pattern was obtained by slitting the stock rectangle at the points indicated by the arrows, chopping the resulting strips, S_1,S_2, and S_3, and then taking the rectangles from S_2 and slitting them again to form the final rectangles. Note than in a process of this type, the widths of the strips that are made by slitting the stock rectangle down its length are no longer necessarily of some ordered width. They may be, for example, of any width $w = \sum_{i=1}^{i=m} a_i w_i$, with a_i nonnegative integers. This means that they may be of virtually any width.

The recursions available for cutting patterns of this sort involve solving a two-stage guillotine problem for every rectangle $w \times L$, where $0 \leqq w \leqq W$, in order to obtain a value $\Pi(w)$ associated with every such w. There then remains still the problem of solving an ordinary knapsack problem in which the number of widths w is quite large in order to determine what strips should be slit from the stock rectangle $W \times L$. When the third-stage cutting is limited in some way, either by the number of cuts that can be made or to patterns involving only one width, the computation is easier.

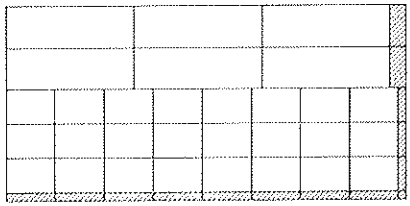

Figure 10

Problems with Many Stock Sizes and Free Two-Stage Cutting

Should several stock rectangles $W_j \times L_j$, $j = 1, \cdots, k$, be available, the linear programming formulation has to be slightly changed to accommodate a new objective function. The changes necessary are exactly those introduced in reference 1 in the formulation of many stock length problems.

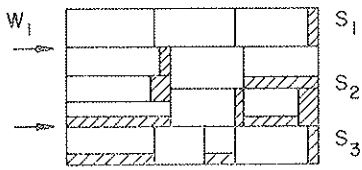

Figure 11

For each j it is necessary to determine a cost C_j for a unit amount of the bar of cross section $W_j \times L_j$. Then in the linear programming formulation the cost coefficient of a column corresponding to a cutting pattern for $W_j \times L_j$ is C_j for any j. The objective function of the knapsack problem should also be changed slightly for we wish now to maximize $\Pi_1 a_1 + \cdots + \Pi_m a_m - C_j$ for $j = 1, \cdots, m$, where for a given j, $[a_1, \cdots, a_m]$ corresponds to a cutting pattern for $W_j \times L_j$. We will maximize the new objective function by maximizing the old for each j subtracting C_j, and maximizing over j.

The solution of the generalized knapsack problem for a k stock rectangle free two-stage cutting problem can be accomplished by the solution of

$k+1$ knapsack problems of type (3). The recursive calculation for one knapsack problem will yield $F_s(L_j)$ for all s and j, and then for each j a recursive calculation is necessary to determine the maximum of $b_1F_1(L_j)+\cdots+b_mF_m(L_j)$ subject to $W_j \geqq b_1w_1+\cdots+b_mw_m$. In the case $L_j=L$ for all j, the latter k recursive calculations reduce to one.

In the glass industry it is sometimes the case that a continuum of stock rectangles is available. This occurs as follows: the stock rectangles are cut from a continuous ribbon of glass and therefore, depending upon whether W or L is the width of the ribbon, there will be stock rectangles for every possible L or every possible W, usually within certain limits. One problem of this kind [(1) below] turns out to be very easy, while the other [(2) below] is very difficult.

(1) Stock Rectangles $W\times L$ with Continuous W

If there are no minimum limits on W then the problem is solvable by one ordinary knapsack calculation as was done in (i) in the section on two-stage guillotine cutting. For by that calculation the maximum worth $F_i(L)$ of every strip of width w_i, $i=1,\cdots,m$, was determined. If the cost of a strip of material of length L and width w_i is cw_i then one need only choose i to maximize $F_i(L)-cw_i$. Consider now the case that W occurs between limits w_0 and W_0. Then following the calculation of step (ii) of the cited section for $W=W_0$, it is necessary to determine the maximum of $F(W)-cW$, for $w_0 \leqq W \leqq W_0$, where $F(W)$ is the optimum value of step (ii).

(2) Stock Rectangles $W\times L$ with Continuous L

This is a very difficult problem requiring the solutions of $|L_0|+1$ knapsack problems, where L_0 is the limiting length and where $|L_0|$ is the number of distinguishable lengths that need be considered. With one knapsack calculation $F_i(L)$ for any L, $L \leqq L_0$, and any i, is determined, but then for each L a step (ii) calculation is necessary.

The Three-Stage Three-Dimensional Problem

The difficulties of extending the methods given above to higher dimensional cutting stock problems are illustrated by considering the three-dimensional problem. Here the over-all problem is to cut small rectangular parallelepipeds out of stock rectangular parallelepipeds. The corresponding knapsack problem is to fit the most valuable combination of parallelepipeds into a parent parallelepiped. The restrictions are similar, the parent parallelepiped is first cut into layers, the layers into strips, and finally the strips into rectangular parallelepipeds. An example of this kind of problem occurs in the cutting up of graphite blocks for anodes.

We can proceed in a manner similar to the two-dimensional case. Suppose that parallelepipeds of dimensions $w_i \times l_i \times h_i$, $i=1, \cdots, m$, must be cut from a stock parallelepiped of dimension $W \times L \times H$ and that Π_i, $i=1, \cdots, m$, are the prices associated with the parallelepipeds for the knapsack calculation. Then, by considering those parallelepipeds that fit into a parallelepiped of dimensions $w_i \times l_j \times H$, we solve a knapsack problem for each i and j to obtain a value $v(w_i,l_j)$ associated with the rectangular end of dimensions $w_i \times l_j$. Once $v(w_i,l_j)$ has been calculated for all i and j, the problem has been reduced to the free two-dimensional one of optimally cutting rectangles $w_i \times l_j$, $i,j=1, \cdots, m$, of price $v(w_i,l_j)$ from a stock rectangle of dimensions $W \times L$.

Again, it is not necessary to do a full recursive calculation for each i and j to obtain all values $v(w_i,l_j)$. A closer inspection shows that, just as in the two-dimensional case, one can fix w_i and then obtain all $v(w_i,l_j)$, for all j, in one recursive calculation. Then for a problem of N different sized parallelepipeds, in which N_1 different widths, N_2 different lengths, and N_3 different heights appear, one would expect to solve the knapsack problem in $2+\min(N_1,N_2,N_3)$ recursive computations, assuming that rotations are not permitted.

Position Dependent Values

In some problems, the worth of a rectangle depends not only on its dimension, but also on its position on the parent rectangle. This is true, for example, if there are defects present or variations of thickness. Let us first take up the one-dimensional problem.

If a piece of length l_i is worth $\Pi_i(x)$ if its right-hand end point is at point x, then the optimal value knapsack of length y contains a value

$$v(y)=\max_i \{\Pi_i(y)+v(y-l_i)\}.$$

This recursion refers the unknown $v(y)$ back to $v(y-l_i)$. If our lengths are multiples of a basic unit, then $v(y)$ can be computed when $v(x)$ is known for $x \leqq y-1$.

The step to two dimensions is a big one. Consider for example the free two-dimensional case when the price for a demanded rectangle is dependent upon its position in the stock rectangle. Even assuming that the rectangles occurring in a strip appear along one edge of the strip does not result in an easy computation. For, in order to reduce the problem to its one-dimensional form it is necessary to calculate the value $\Pi_i(y)$ of a strip of width w_i for each possible value of y, $w_i \leqq y \leqq W$, where y is the location of the upper edge of the strip in the stock rectangle. Then one recursive computation will provide $\Pi_i(y)$ for fixed y and all i, but $|W-w_{min}|+1$ such computations are necessary to compute $\Pi_i(y)$ for all i

and y. With this data calculated the problem can be solved by a recursion like the one-dimensional recursion above.

Important simplifications are possible if the position dependent values are a consequence of well defined defects. A practical illustration, for example, is the problem of cutting defect-free pieces for furniture of given widths and lengths from lumber with defects. In this case a further simplification occurs since the patterns permitted in a strip, which in this case runs across a board rather than down its length, are simply multiples of a single given width.

THE CORRUGATED BOX PROBLEM

In addition to the problem of generating the cutting patterns and determining the number of times each is to be used, which is the aspect of the problem we have discussed so far, there are often special side conditions to be met or sequencing problems involving the cutting operations that must be solved. Sometimes these are relatively unimportant; sometimes they dominate the problem and reduce the question of pattern generation to relative insignificance. We will illustrate one of this latter type in the next section after discussing in this section in some detail the corrugated box problem.

Corrugated boxes are made from rectangles of corrugated paper. The paper is generally manufactured and cut immediately on one machine. The corrugated paper is manufactured from rolls of paper so that the length dimension of the stock can be ignored. As the corrugated paper moves off the corrugator, it is immediately slit in the length and chopped across the width. Generally, however, there are but two chopping knives available, an upper and lower, each of which has its own limitations as to the length it can chop. Consequently, the problem is a two-dimensional problem in which the strips are grouped into two groups for chopping but in which the effectively infinite length of the stock rectangle trivializes the chopping patterns in a strip to simple repetitions of a demanded length.

If one assumes that for each i and j, if $w_i \neq w_j$ then $l_i \neq l_j$, and one is restricted to cutting exactly, then the problem immediately reduces to a one-dimensional problem. This we call the restricted corrugated box problem. For this problem at most two distinct widths may occur in a cutting pattern. The demand d_i for a rectangle of dimensions $w_i \times l_i$ becomes, in the one-dimensional problem, a demand for a quantity $d_i l_i$ of paper of width w_i. The cutting patterns that are in the solution to this problem are the slitting patterns for the corrugated box problem, since each such pattern contains at most two widths and therefore the strips slit can be grouped into two groups for chopping.

The assumptions made for the restricted corrugated box problem are,

in our experience, realistic. Methods for solving the corrugated box knapsack problem when these assumptions are not made can be devised but as we are interested in discussing an important side condition of the problem, we will leave a discussion of these methods to a later paper devoted to the knapsack problem.

The only special feature of the restricted corrugated box knapsack problem apart from the restriction of at most two widths to a slitting pattern is the form of the cost of the stock widths. Generally, there are a large number of different paper widths available for a corrugator—ten or even twenty is not unusual. The purpose of maintaining the large number of widths is of course to reduce paper waste. But a reduction of paper waste is not the only objective in scheduling a corrugator for one must also try to maintain an efficient utilization of the corrugator—to run a 60″ width of a paper on an 80″ corrugator represents only a 75 per cent utilization of the machine.

The two objectives of keeping paper waste low and machine utilization high can be properly combined into one minimum cost objective if proper costs are assigned to the stock widths. If the cost assigned to a stock width W is to be the cost of 1000 lineal feet of corrugated paper of that width, then its cost is determined by the formula C_0+cW, where C_0 is the cost of operating the corrugator long enough to produce 1000 feet of paper and c is the cost of the paper and glue in 1000 lineal feet of paper one inch wide, if W is measured in inches.

There are several different ways of solving a knapsack problem in which at most two widths may occur in any slitting pattern and in which there are many stock widths $W_j, 1 \leqq j \leqq r$, each with a cost C_0+cW_j. The simplest, although not necessarily the most efficient, is to simply enumerate all possible slitting patterns by generating all possible pairs of nonnegative integers α_i and α_j, where $1 \leqq i < j \leqq m$ and $\alpha_i w_i + \alpha_j w_j \leqq W_s$, for $s = 1, \cdots, k$. As each pair (α_i, α_j) is generated its value $v(\alpha_i, \alpha_j) = \alpha_i \Pi_i + \alpha_j \Pi_j$ is compared with the previously largest value obtained for W_s and this value is updated if necessary. Other methods for this problem will be discussed in the forthcoming paper devoted to the knapsack problem.

SCHEDULING THE TRIPLEX IN THE CORRUGATED BOX PROBLEM

THE SIDE condition we would like to discuss concerns the sequencing of the slitting patterns obtained as the solution to the corrugated box cutting stock problem. To understand this problem, it is necessary to understand that the slitting of the paper on the corrugator is done by circular knives mounted on an axle at the output end of the corrugator. At the same time the paper is slit it is also creased by creasers mounted on the same axle as the knives, the creased lines being lines of fold for the corrugated boxes.

After being slit and creased, the paper goes to the upper and/or lower chopping knives to be cut into rectangles.

In order to make it unnecessary to stop the corrugator for the resetting of knives and creasers, generally three axles of knives and creasers are mounted on a rotatable frame that permits two axles to be set while the third is being used. It is therefore desirable and important to so sequence the slitting patterns that the corrugator is stopped as infrequently as possible for the resetting of the knives.

A precise solution to this sequencing problem is not meaningful because the data available about how long it takes for a man to set the knives and creasers for a given slitting pattern is inaccurate, and because of changes in the speed of the corrugator, the length of time it takes for a corrugator to complete one slitting pattern, cannot be known accurately. Because of this, simplifying assumptions regarding the scheduling of the corrugator are not unreasonable. The assumptions we make are:

1. The corrugator runs at a constant speed so that a unit of time can be translated into a unit of length of paper run through the corrugator.

2. The time needed to set the knives and creasers of the axle for one slitting pattern is the same for all slitting patterns.

Assumption 2 is not generally true since the number of slitters and creasers that must be set on an axle affects the time for setting an axle. However, it does provide an approximation. Without this assumption, the problem of scheduling the slitting pattern is unsolved except in the case where only two axles of knives and creasers are available, as discussed in references 8 and 9; in 3 below we assume there are the usual three.

Assumptions 1 and 2 mean that for some given λ, the time needed to set the axles for one slitting pattern is the time needed for λ units of length of paper to be run through the corrugator.

3. There are three axles of knives and creasers available, two of which can be set while the third is being used.

A fourth assumption is also made, but some preliminary discussion is necessary for its introduction.

Let the lengths of the runs of the slitting patterns to be scheduled be $S_1, S_2, \cdots, S_m$. For each of these numbers S_j there exists an integer $q(S_j)$ and a nonnegative $r(S_j)$ such that

$$S_j = \lambda \cdot q(S_j) + r(S_j), \quad \text{where} \quad 0 \leqq r(S_j) < \lambda.$$

$q(S_j)$ is the number of complete axles that can be set during the time it takes the corrugator to produce the S_j units of paper for the jth slitting pattern. A slitting pattern for which $q(S_j) = n$ will be called a q_n pattern.

If $q(S_j) \geqq 1$ for every j then no scheduling problem exists because no

matter in what order the slitting patterns are scheduled, it is always possible to set the axle for the succeeding pattern during the running of the preceding pattern. However, if there are q_0 patterns, there appears the difficulty that the axles for the succeeding pattern of any one of them might not be set and the corrugator would have to be stopped.

The only way that the q_0 patterns can be accommodated is to set in advance the axles of patterns that must succeed them. For example, if S_j is a q_2 pattern, S_{j+1} a q_0, and S_{j+2} a q_1, then the sequence

$$\cdots, S_j, S_{j+1}, S_{j+2}, \cdots$$

allows enough time for the setting of the axles for S_{j+1} and S_{j+2} since they can both be set during the running of S_j.

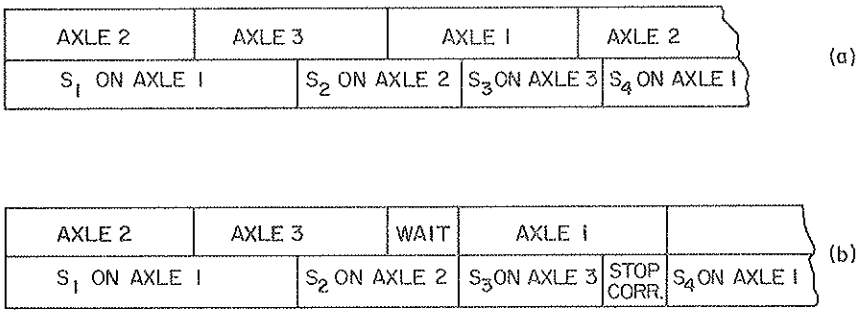

Figure 12

There are also situations in which part of an axle is set during one pattern run and part during the next; for example, if $r(S_j)+r(S_{j+1}) \geqq \lambda$, then the sequence

$$\cdots, S_j, S_{j+1}, \cdots$$

is satisfactory if S_j is a q_1 and S_{j+1} is a q_0. For, if we enter the run of S_j with one axle set (the one with the pattern for S_j), we will finish S_j with S_{j+1}'s axle set and part of S_{j+2}'s. During the run of S_{j+1} we will, because $r(S_j)+r(S_{j+1}) \geqq \lambda$, finish the axle for S_{j+2} and so be ready to continue.

There are, however, reasonable limits that should be set on this process of setting more than one partial axle during the running of a single slitting pattern. This is because there is an interruption and loss of time when the triplex is rotated and repositioned to change patterns—often the man who sets the axles also repositions the triplex. Therefore, we make a final assumption to prohibit too much of this during short runs.

4. If a pattern is a q_0 or a q_1, not more than one partial axle can be set during its run.

If we draw a figure giving run times on one line and set-ups on another, then because of 4, Fig. (12a) shows a prohibited arrangement since during the running of S_2, a q_0, work is performed on both axle 3 and axle 1; while Fig. (12b) is a satisfactory arrangement.

Because of assumption 3 that there are but 3 axles, a q_n pattern, where $n \geqq 2$, can be assumed to satisfy a stronger assumption than 4; namely, that no partial axles are set during its running, but only two complete axles. Hence there is no loss in assuming that every q_n pattern, where $n \geqq 2$, is a q_2 pattern. Further, the same assumption ensures that an axle is not set during the running of more than two different patterns as in Fig. 13, since that would require at least 4 axles.

From 3 and 4 we have therefore

5. (*a*) During the running of a q_n pattern, where $n \geqq 2$, at most two complete axles can be set and no partial axles need be set.

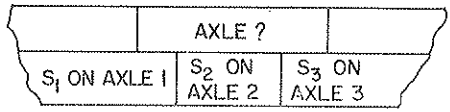

Figure 13

(*b*) Any axle that is set during the running of more than one pattern is set during the running of a pair of patterns consisting of (i) two q_0's; or (ii) two q_1's; or (iii) a q_0 and a q_1.

With these assumptions, we will take up the problem of optimal sequencing. A preliminary remark is helpful. Suppose, for purposes of exposition, we consider instead of assumption 4 a much stronger one. Assume (temporarily) that axles can only be set up entirely within a run; in other words, *no* overlapping at all of either kind shown in Fig. 12 is permitted. Then the sequencing problem becomes trivial. For—assuming that one starts with one axle set—one can run all the q_1's, each setting an axle for its successor. There then remain the q_0's and the q_2's. The only problem is the q_0's, and the only way a q_0 can ever be run (under the no overlap restriction) is immediately following a q_2, for, since nothing is set up during the q_0 run, two axles, one for the q_0 and one for its successor, must be ready at the beginning of a q_0 run. So q_0's can only be run in sequences like

$$\cdots, q_2, q_0, q_2, q_0, \cdots$$

with each q_2 taking care of one q_0. Therefore, under the strong no overlap assumption, the number of unset or unschedulable axles is $N_0 - N_2$, where N_i is the number of q_i.

Now let us return to our actual assumption **4**. In any sequence satisfying **5**, there is a natural pairing between those q_0 and q_1 patterns that share the setting of an axle. Because of assumption **4**, the pairing is well defined; i.e., a pattern is paired with at most one other pattern. Further, once the pairings are determined and fixed, the problem reverts to the strong no overlap situation in which the existing pairs are juggled around like the individual patterns under a no overlap assumption. The no overlap assumption then holds because all overlaps are contained *within* pairs. We will now discuss in greater detail the material just sketched out.

First, we observe that there is no point in pairing S_i with S_j unless the total number of axles set in the pair is greater than $q(S_i)+q(S_j)$, since if we only have equality the excess of S_i+S_j over $q(S_i)+q(S_j)$ cannot be utilized. The condition for this is, of course, that $r(S_i)+r(S_j)\geqq\lambda$. We will consider only these pairings from this point on.

The only cases that can arise are:

(i) a q_0-q_0 pair; i.e., $q(S_i)<\lambda$, $q(S_j)<\lambda$, and $r(S_i)+r(S_j)\geqq\lambda$. Note that a q_0-q_0 pair sets one axle, and uses up two settings. Therefore, for purposes of sequencing, a q_0-q_0 pair is equivalent to a single q_0.

(ii) a q_0-q_1 or q_1-q_0 pair. This pair uses up two axles and sets two; for sequencing purposes, it is equivalent to a single q_1.

(iii) a q_1-q_1 pair. This uses up two axles and sets three; it is equivalent to a single q_2.

Once the pairings have been decided upon, then we can replace the pairs by their equivalences (a q_0-q_1 by a q_1, etc.), and then sequence the new problem under the no overlap assumption, since all overlaps have already been taken into account. The merit of the final schedule is measured by the number of q_0 axles that can be set in the schedule either singly or in q_0-q_1 pairs.

Clearly then, the whole problem is to create the proper pairing. We will show that the following procedure gives a pairing that results in a minimal number of unset axles.

1. The q_1 patterns and the q_0 patterns on the list are listed together in increasing size of r number.

2. A q_1 pattern on the list, say with r number r_1, is combined with the first q_1 or q_0 pattern on the list with r number r_2 such that $r_1+r_2\geqq\lambda$. When a pair is formed, the pair is struck from the list, while if no pair is formed, the q_1 pattern is struck from the list.

Step **2** is repeated until no q_1 patterns remain on the list.

3. A q_0 pattern on the list, say with r number r_1, is combined with the first q_0 pattern with r number r_2 such that $r_1+r_2\geqq\lambda$. When a pair is formed, the pair is struck from the list, and when a pair is not formed, the q_0 pattern is struck from the list.

Step 3 is repeated until no q_0 patterns remain on the list.

We have now completed the pairing; the actual sequencing as outlined above is done by Step 4.

4. The q_0 pairs formed in 3 are assigned to q_2 patterns and q_1-q_1 pairs formed in 2. Should all q_0 pairs be so assigned, then q_0 patterns that are not members of pairs formed in 2 or 3 are assigned to any remaining q_2 patterns and q_1-q_1 pairs.

We will now show that this algorithm sets the largest number of q_0 patterns when we say that a q_0 pattern is set by a schedule if it immediately follows, either singly or in a q_0-q_0 pair, a q_2 or q_1-q_1 pair.

Now let B be a schedule which sets the optimal number of q_0 patterns and let B' be a schedule produced by the algorithm. As we remarked earlier, any schedule satisfying the assumptions 3 and 4 produces a pairing of some of the q_0 and q_1 patterns, and in particular, therefore, B produces such a pairing. We will show first that all q_1 patterns occurring in pairs in B' can occur in exactly the same pairs in some optimal B. For let S' be the first q_1 pattern occurring in a pair (S',T') in B' which does not occur in that pair in B, considering the patterns in the order in which they have been considered in B'.

Case (A). (S',T) and (S,T') are both pairs in B. Since S' is the first q_1 pattern not occurring in B in the same pair in which it occurs in B', T must have been available when S' was being paired so that necessarily $r(T)\geqq r(T')$, and therefore (S,T) is a possible pair; that is, $r(S)+r(T)\geqq\lambda$. In each of the following cases, we will show that the pairs (S',T) and (S,T') in B can be replaced by the pairs (S,T) and (S',T') without affecting the merit of the schedule.

(i) S and T are both q_1's. If T' is a q_0 then a q_1-q_1 pair (S',T) and a q_1-q_0 pair (S,T') are replaced by a q_1-q_0 pair (S',T') and a q_1-q_1 pair (S,T). Since any pair is interchangeable with any other pair of the same kind, the new schedule has the same merit as B. Similarly, if T' is a q_1.

(ii) S is a q_1 and T is a q_0. If T' is a q_0 then two q_1-q_0 pairs are replaced by two others leaving the merit of B unchanged. If T' is a q_1 then a q_1-q_0 pair and a q_1-q_1 pair are replaced by other pairs of the same kind.

(iii) S is a q_0 and T is a q_1 is similar to (ii).

(iv) S and T are both q_0's. If T' is a q_0 then a q_1-q_0 pair and a q_0-q_0 pair are replaced by other pairs of the same kind. If T' is a q_1 then two q_1-q_0 pairs are replaced by a q_1-q_1 pair and a q_0-q_0 pair. However in B the q_1-q_0 pairs are like q_1's standing alone; therefore removing them from the schedule does not affect the remaining schedule. Further a q_1-q_1 pair can be followed by a q_0-q_0 pair so that the merit of the new schedule is unchanged from B.

Case B. (S',T) is a pair in B but T' remains unpaired. In each of the following cases we will show that the pair (S',T) and the pattern T' can

be replaced in B by the pair (S',T') and the pattern T without affecting the merit of B.

(i) T is a q_0. If T' is also a q_0 then a q_1-q_0 pair and a q_0 pattern are replaced with another q_1-q_0 pair and q_0 pattern. If T' is a q_1 then a q_1-q_0 pair and a q_1 pattern are replaced by a q_1-q_1 pair and a q_0 pattern. However the q_0 pattern T can follow the q_1-q_1 pair (S',T') so that the merit of B is unchanged.

(ii) T is a q_1. If T' is a q_0 then a q_1-q_1 pair and a q_0 pattern are replaced by a q_1-q_0 pair and a q_1 pattern. The merit of B can only be decreased if the q_1-q_1 pair (S',T) was followed by a q_0 or a q_0-q_0 pair and the q_0 pattern T' were preceded by either a q_2 or a q_1-q_1 pair. In this case, without loss of merit of B, a q_0 or a q_0-q_0 pair following (S',T) can be interchanged with T' so that T' can be assumed to follow (S',T) in B. But a q_1-q_1 pair followed by a q_0 can be replaced by a q_1-q_0 pair followed by a q_1 without loss of merit. If T' is a q_1 the result is immediate.

Case C. When S' is unpaired and (S,T') is a pair in B is similar to case B.

Case D. Neither S' nor T' are paired in B. Then we must show for each of the following cases that S' and T' can be paired in without affecting the merit of B.

(i) T' is a q_0. Certainly T' can be paired with S' without decreasing the merit of B, for pairing S' and T' can only release a q_2 or q_1-q_1 pair which preceded T' in B.

(ii) T' is a q_1. Then no q_0's can be unset in B since otherwise the q_1-q_1 pair (S',T') followed by an unset q_0 of B would improve the schedule B. Hence if any q_0's occur at all they are preceded by q_2's or q_1-q_1 pairs, which could equally well be replaced by (S',T').

We have shown that any pair in B' containing a q_1 can occur in an optimal schedule B. Since no further q_1's can be paired, apart from those occurring in pairs in B', we have shown that the pairs involving q_1's in B can be assumed to be exactly those of B'. So we can now assume that B and B' have exactly the same q_0 patterns available for forming q_0-q_0 pairs. Hence it is only necessary now to show that the algorithm produces the optimum number of q_0-q_0 pairs, since such a pair requires the same support as a q_0. But let S' be the first q_0 pattern not paired in B as it is in B', considering the patterns in the order in which they are considered in B', and let T' be its mate in B' while T is its mate in B. Since $r(T)\geqq r(T')$, T and T' can be interchanged. We have therefore shown that the pairing and sequencing described in Steps 1–4 will produce the smallest number of unset axles.

Should an optimal schedule still leave axles unset, a slight relaxing of the ground rules for scheduling may assist. For the effect of assumption

3 is seen in 5(a) to result in a wasting of potential set-up time since a q_n, where $n \geqq 3$, is regarded only as a q_2. However if a q_n pattern, where $n \geqq 3$, is split into two or more patterns, and the fact that there are exactly three axles available is exploited, some of the otherwise wasted set-up time can be utilized. For example, if $q(S)=3$, then the slitting pattern can be run in two parts, one of length 2λ and the other of length $\lambda+r(S)$. While the first part of the pattern is being run with the first axle, the remaining two axles can be set. While these two axles are being run, nothing need be done because when they have been used, the first axle is still available for the second part of the slitting pattern. In this fashion, it is possible to set up q_0 patterns by taking a pattern running for a length S for which $q(S)>2$, and splitting it up into patterns $T_1, \cdots, T_k$, where $T_i=2\lambda$ and T_k is the remainder upon dividing S by 2λ, and then sandwiching the q_0's in pairs between successive T_i's. When a q_n pattern S, $n \geqq 3$, has been so split and used, the whole group formed is defined to be a $q_{n'}$ pattern with remainder $r(S)$ and with $q_{n'}$ 0 or 1, depending upon whether n is even or odd respectively.

GROUPING TO INCREASE AVERAGE SLITTING PATTERN RUN

In some cases, it is possible to predict that for a given λ a schedule based on the assumptions 1 to 4 will not be possible. For if there are m different rectangles ordered, then in general there will be m slitting patterns necessary to fill the orders. Hence, using even the smallest width of paper available and calculating total length of run from the total area of paper ordered, it can occur that the average slitting pattern run is less than λ. Consequently, no schedule that will set all q_0 patterns can result. The only way that the average slitting pattern run can be increased is by decreasing the number of slitting patterns used to fill the orders. In a one-dimensional cutting stock problem, it is possible to reduce the number of slitting patterns appearing in a solution by accepting a less than optimal solution. This is accomplished by what we call the grouping of widths. Two ordered widths w_1 and w_2, $w_1>w_2$ with demands N_1 and N_2 can be replaced by the single width w_1 with a demand N_1+N_2 if a second trimming to size is done after the slitting. For the N_2 units of width w_2 required are met by trimming that amount of the N_1+N_2 units of w_1 produced and thereby producing a necessary additional waste of $(w_1-w_2)\cdot N_2$ square units. But one fewer ordered widths occur and, as a result, one fewer slitting patterns can be expected.

For the corrugated box problem, if all the orders have to be filled, and the average slitting pattern run is less than γ, then grouping is a means for raising the average. The costs associated with a given grouping may not consist only of the necessary additional waste, for a second operation, per-

formed simultaneously with slitting on the corrugator, is that of creasing and widths that are grouped cannot be creased on the corrugator. Consequently, grouping widths can result in additional operations off the corrugator. However, since small orders are generally creased off the corrugator anyway, confining the grouping to small orders results in an increased average slitting pattern run at the cost of necessary additional waste.

If m^* is the desired number of slitting patterns to provide an average run of λ, then $m-m^*$ widths must be put into groups in which they are not the largest width. The grouping problem presenting itself is then to group sufficiently many small orders so as to achieve the number m^* with the least possible necessary additional waste. This is a problem readily solvable by a simple recursive calculation. For let $w_1, \cdots, w_s$ be the widths that may be grouped listed in decreasing order of size with demands respectively $N_1, N_2, \cdots, N_s$. The problem is then to choose $r=s-(m-m^*)$ widths $w_{i_1}, w_{i_2}, \cdots, w_{i_r}$, $i_1=1<i_2<\cdots<i_r\leqq s$, assuming $s>(m-m^*)$, such that the necessary waste resulting from the groups $\{w_1, \cdots, w_{i_2-1}\}$, $\{w_{i_2}, \cdots, w_{i_3-1}\}$, $\cdots$, $\{w_{i_r}, \cdots, w_s\}$, namely

$$T=\sum_{i=i_1}^{i_2-1}(w_{i_1}-w_i)\cdot N_i+\sum_{i=i_2}^{i_3-1}(w_{i_2}-w_i)\cdot N_i+\cdots+\sum_{i=i_r}^{i=s}(w_{i_r}-w_i)\cdot N_i,$$

is as small as possible.

Let T_{uv}, where $1<v\leqq s$, and $1\leqq u\leqq v$, be the least possible necessary additional waste that can result if the widths $w_1, w_2, \cdots, w_v$ are grouped into u groups, and let $\alpha(u,v)$ be an integer v', $u\leqq v'\leqq v$, such that there is a grouping of the widths with T_{uv} as its necessary waste and with $\{w_{v'}, \cdots, w_v\}$ as its last group. These functions can be recursively defined as follows:

For $1\leqq v\leqq s$, $T_{v1}=\sum_{j=1}^{j=v}(w_1-w_j)\cdot N_j$ and $\alpha(1,v)=1$.
For $2\leqq u\leqq v\leqq s$, $T_{uv}=\min\{T_{u-1v'}+\sum_{i=v'+1}^{i=v}(w_{v'+1}-w_i)\cdot N_i; u-1\leqq v'<v\}$

and $\alpha(u,r)=v'+1$, where v' is the index realizing the minimum T_{uv}; i.e., v' is such that $u-1\leqq v'<v$ and

$$T_{u-1v'}+\sum_{i=v'+1}^{i=v}(w_{v'+1}-w_i)\cdot N_i=T_{uv}.$$

The least possible necessary additional waste resulting from grouping the s widths into r groups is T_{rs} and this waste is achieved by the grouping with 'lead' widths $w_{i_1}, w_{i_2}, \cdots, w_{i_r}$ determined by the recursive equation: $i_r=\alpha(r,s)$ and $i_{t-1}=\alpha(t-1,i_t-1)$, $t=r, r-1, \cdots, 2$, where necessarily $i_1=1$.

REFERENCES

1. P. C. Gilmore and R. E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem," *Opns. Res.* **9**, 849–859 (1961).

2. ——— AND ———, "A Linear Programming Approach to the Cutting Stock Problem, Part II," *Opns. Res.* **11**, 863–888 (1963).
3. L. V. KANTOROVITCH, "Mathematical Methods of Organizing and Planning Production," reprinted in *Management Sci.* **6**, 366–422 (1962).
4. P. F. REITH, "The Trim Problem," a report issued by IBM, Amsterdam.
5. GEORGE B. DANTZIG, "Discrete Variable Extremum Problems," *Opns. Res.* **5**, 266–276 (1957).
6. R. BELLMAN, "Some Applications of the Theory of Dynamic Programming—a Review," *Opns. Res.* **2**, 275–288 (1954).
7. GEORGE B. DANTZIG AND PHILIP WOLFE, "Decomposition Principle for Linear Programs," *Opns. Res.* **8**, 101–111 (1960).
8. P. C. GILMORE AND R. E. GOMORY, "A Solvable Case of the Travelling Salesman Problem," *Proc. Nat. Acad. Sci.* **51**, 178–181 (1964).
9. ——— AND ———, "Scheduling a One-Stage Variable Machine: A Solvable Case of the Travelling Salesman Problem," *Opns. Res.* **12**, 655–679 (1964).