Reprinted from OPERATIONS RESEARCH
Vol. 9, No. 6, November-December, 1961
Printed in U.S.A.

A LINEAR PROGRAMMING APPROACH TO THE CUTTING-STOCK PROBLEM

P. C. Gilmore and R. E. Gomory

International Business Machines Corporation, Research Center, Yorktown, New York

(Received May 8, 1961)

The cutting-stock problem is the problem of filling an order at minimum cost for specified numbers of lengths of material to be cut from given stock lengths of given cost. When expressed as an integer programming problem the large number of variables involved generally makes computation infeasible. This same difficulty persists when only an approximate solution is being sought by linear programming. In this paper, a technique is described for overcoming the difficulty in the linear programming formulation of the problem. The technique enables one to compute always with a matrix which has no more columns than it has rows.

SOME linear programming problems arising from combinatorial problems become intractable because of the large number of variables involved. Usually each variable represents some activity, and the difficulty is that there are too many possible competing activities satisfying the combinatorial restrictions of the problem. An example of this is the cutting-stock problem described below in a form similar to that used by EISEMANN.[1]

The purpose of this paper is to point out that this difficulty can be overcome by a method basically identical with the idea that can be considered as implicit in references 2 and 3, and which is essentially this. When, in the simplex method, we reach the stage of 'pricing out' or looking for a new column or activity that will improve the solution, instead of looking over a vast existing collection of columns to pick out a useful one, we simply create a useful column by solving an auxiliary problem. In reference 2 the problem is a shortest-path problem, in reference 3 a problem in linear programming.

In the problem considered here the auxiliary problem will be of the integer programming variety, but of such a special type (the 'knapsack' type) that it is solvable by several methods (*see* reference 4). If the same technique were applied to the problems discussed in reference 5, the calculation of WAGNER AND WHITIN[9] would be applicable to the auxiliary problem, while for the problem discussed in reference 6 a general integer programming technique such as discussed in references 7 or 8 would presumably be required.

Turning to the cutting-stock problem we assume that a stock of standard lengths $L_1, L_2, \cdots, L_k$ of one material is maintained from which one is to cut lengths to fill incoming orders. An unlimited number of pieces are assumed available in stock for each of the stocked lengths $L_1, L_2, \cdots, L_k$. An order consists of a request for a given number N_i of pieces of length ℓ_i of the stocked material, for $i=1, 2, \cdots, m$. As long as for some j and all i, $L_j \geqq \ell_i$, an order can be filled. A cost is assigned to each of the stocked lengths and the cost of filling an order is simply the total cost of the stock material cut to fill the order. The problem is to fill the orders from stock at the least cost.

By an activity, we will mean the cutting of a specified stock length in a specified manner. Thus, for example, the cutting from a stock length 17 of three pieces, one of length 5 and two of length 4, is an activity. By assigning a variable to each of the possible activities that cut ordered lengths $\ell_1, \cdots, \ell_m$ from stock lengths $L_1, \cdots, L_k$, the cutting stock problem can be posed as an integer linear programming problem, where the value taken by a variable indicates the number of times the activity is to be engaged in. The variables $x_1, \cdots, x_n$ assigned to activities must satisfy m inequalities:

$$a_{i1}\,x_1+a_{i2}\,x_2+\cdots+a_{in}\,x_n \geqq N_i, \qquad (i=1, \cdots, m)$$

if an order for N_i pieces of length ℓ_i is to be filled, where a_{ij} is the number of pieces of length ℓ_i created by the jth activity. The cost function to be minimized is then

$$c_1\,x_1+c_2\,x_2+\cdots+c_n\,x_n, \tag{1}$$

where c_i is the cost of the stock length cut by the ith activity. Introducing slack variables $x_{n+1}, \cdots, x_{n+m}$, the cutting stock problem can be described as the problem of finding integers $x_1, \cdots, x_{n+m}$ satisfying

$$a_{i1}\,x_1+\cdots+a_{in}\,x_n-x_{n+i}=N_i, (i=1, \cdots, m) \tag{2}$$

and

$$x_j \geqq 0, \qquad (j=1, \cdots, n+m) \tag{3}$$

for which (1) is a minimum.

There are two factors contributing to making this formulation of the cutting-stock problem impractical. First is the size of n, which can be enormous when the number k of stock lengths and the number m of requested lengths is any reasonable size. Second is the restriction to integers.

Consider the second factor first. If the restriction were removed, then a solution to the cutting-stock problem would in general be noninteger. Given a noninteger solution there are several traditional ways that one can determine an integer approximate solution; for example, one can round up to the nearest integer adding necessarily to the cost or one can round down

to the nearest integer and treat the filling of the created unfilled portion of the order as a separate problem to be solved by *ad hoc* methods. If the noninteger values are large, the fractional change in the cost caused by rounding out will usually be small. Since the cost first obtained is the smallest possible with or without the restriction to integers, a small increase in it can often be tolerated even though the resulting cost may not be the least possible attainable with integers. We will, at any rate, only consider in this report the linear programming solution of the cutting-stock problem in which the variables are not restricted to be integer, since our purpose is the description of an efficient method for dealing with the first factor, the very large number n of variables.

It is worth noting that the removal of the restriction to integers on the variables allows one to drop the slack variables from equation (2), since for any solution of (2) and (3) in which slack variables take positive values there exists a solution with the same cost in which no slack variables take a positive value. For let there be a solution $(\bar{x}_1, \cdots, \bar{x}_n, \bar{x}_{n+1}, \cdots, \bar{x}_{n+m})$ to (2) and (3) for which $\bar{x}_{n+1} \neq 0$. We can assume that for some i, $a_{1i}\,\bar{x}_i \geqq \bar{x}_{n+1}$; that is, that the ith activity contributes in the solution at least as much to the order for length ℓ_1 as the order has been over-fulfilled. For if there is no such i, let the jth variable be the first taking a nonzero value $\bar{x}_j$ and let the kth activity be the activity that is identical with the jth in all respects except that it does not create any pieces of length ℓ_1; that is, the pieces of length ℓ_1 created by activity j are regarded in activity k as scrap. Then another solution $(\bar{x}_1', \cdots, \bar{x}_n', \bar{x}_{n+1}', \cdots, \bar{x}_{n+m}')$ of (2) and (3) with the same cost as the original is obtained by taking $\bar{x}_i' = \bar{x}_i$, for $i \neq j$, k, $n+1$, $\bar{x}_j' = 0$, $\bar{x}_k' = \bar{x}_k + \bar{x}_j$, and $\bar{x}_{n+1}' = \bar{x}_{n+1} - a_{1j}\,\bar{x}_j$ since the cost coefficients of x_j and x_k are identical. In this new solution the value of the slack variable x_{n+1} has been reduced. If it has not been reduced enough to give $a_{1i}\,\bar{x}_i' \geqq \bar{x}_{n+1}'$ for some i, then the above process is repeated until a solution is found for which one variable does satisfy this inequality. But if $a_{1j}\,\bar{x}_j \geqq \bar{x}_{n+1}$ then the slack variable x_{n+1} can be given a zero value in a solution with the same cost as the given solution. For let the kth activity, as above, be the activity which is identical with the jth in all respects except that it does not create any pieces of length ℓ_1 and define the new solution $(\bar{x}_1', \cdots, \bar{x}_n', \bar{x}_{n+1}', \cdots, \bar{x}_{n+m}')$ by taking $\bar{x}_i' = \bar{x}_i$ for $i \neq j$, k, $n+1$, $\bar{x}_j' = \bar{x}_j - (\bar{x}_{n+1})/a_{1j}$, $\bar{x}_k' = \bar{x}_k + (\bar{x}_{n+1})/a_{1j}$, and $\bar{x}_{n+1}' = 0$. Since the cost coefficients are, as before, identical for x_j and x_k, the new solution has a cost identical with the previous solution.

Although the slack variables can be dropped when the restriction to integers is removed, there may be advantages to not dropping them. For without the slack variables every minimal solution to the problem will in general be in terms of exactly m activities, while with the slack variables a

minimal solution may be in terms of less than m activities. It is therefore possible when one considers the final rounding out to an integer solution that the solution obtained with the use of slack variables will be better than one obtained without. We will in any case describe a computation routine for the problem with slack variables that can be modified to a routine for the problem without slack variables by dropping one step.

The simplex computational procedure when used to determine a solution of (2) subject to (3) for which (1) is a minimum provides for any given basic feasible solution of (2) and (3) a successor basic feasible solution for which the value of (1) is less than for the given solution.

In particular if a basic solution of (2) and (3) is given, the simplex procedure tests each of the other variables in turn until one is found that can replace one of the current basic variables. Let us assume that the variables in a given basic feasible solution are $x_1, x_2, \cdots, x_m$. Let $\boldsymbol{P}_i$ be the vector $(a_{1i}, a_{2i}, \cdots, a_{mi})$ and c_i the cost coefficient in (1) associated with the variable x_i; thus if x_i is a slack variable then c_i is 0 and the vector has a single nonzero coordinate -1. Let $\boldsymbol{P} = (a_1, a_2, \cdots, a_m)$ define an undetermined new activity that cuts from a stock length L having a cost c. Further let A be the matrix with $\boldsymbol{P}_1, \cdots, \boldsymbol{P}_m$ as columns. Since $\boldsymbol{P}_1, \cdots, \boldsymbol{P}_m$ form a basis there is the usual column vector $\boldsymbol{U}$ satisfying the equation

$$\boldsymbol{A} \cdot \boldsymbol{U} = \boldsymbol{P}, \tag{4}$$

and the new activity can be used in a solution that will be an improvement over the given solution if and only if

$$\boldsymbol{C} \cdot \boldsymbol{U} > c \tag{5}$$

where $\boldsymbol{C}$ is the row vector with coefficients $c_1, c_2, \cdots, c_m$. Hence if the row vector $\boldsymbol{C} \cdot \boldsymbol{A}^{-1}$ has coefficients $b_1, \cdots, b_m$ then from (4) and (5) can be concluded that there exists an activity cutting from L that can profitably be used if and only if there exist nonnegative *integers* $a_1, \cdots, a_m$ satisfying

$$L \geqq \ell_1 a_1 + \cdots + \ell_m a_m, \tag{6}$$

and

$$b_1 a_1 + \cdots + b_m a_m > c. \tag{7}$$

It is of course important that $\boldsymbol{C} \cdot \boldsymbol{A}^{-1}$ is always on hand as a part of the normal simplex computational procedure.

One method of determining whether there exist positive integers a_i satisfying (6) and (7) would be to determine nonnegative integers satisfying (6) for which $b_1 a_1 + \cdots + b_m a_m$ is a maximum, for if such integers did not satisfy (7) then none would. Hence the problem of choosing a new

variable in the simplex procedure for the cutting-stock problem can be expressed as the problem of finding a solution for up to k auxiliary problems (one for each of the stock lengths $L_1, \cdots, L_k$) each one of which is an integer linear programming problem. We will show that these k auxiliary problems can be solved by a *single* dynamic programming computation or in some cases by an even more rapid *ad hoc* method.

Since the problem of maximizing $b_1 a_1 + \cdots + b_m a_m$ subject to (6) is a generalization of the knapsack problem, it can be solved by dynamic programming in a manner very similar to that described by DANTZIG in reference 4. Defining $F_s(x)$ to be the maximum of $b_1 a_1 + \cdots + b_s a_s$ subject to the inequality $x \geqq \ell_1 a_1 + \cdots + \ell_s a_s$, then

$$F_{s+1}(x) = \max_r \{r b_{s+1} + F_s(x - r\ell_{s+1})\},$$

where r need only be chosen such that $0 \leqq r \leqq [x/\ell_{s+1}]$, and square brackets are used to denote the largest integer part of the argument appearing within them. That only one complete dynamic programming computation is necessary in order to introduce a new variable in the simplex procedure can be readily seen, for if say L_1 is the largest of the stock lengths then in the course of computing $F_m(L_1)$ one has automatically also computed $F_m(L_2), \cdots, F_m(L_k)$.

But even this amount of computation will frequently be more than is necessary since one need only find some $a_1, \cdots, a_m$ satisfying both the inequalities (6) and (7) when L is taken to be one of the stock lengths $L_1, \cdots, L_k$ and c is taken to be the cost of the stock length. Thus, any simple *ad hoc* method of solution may be used until the method does not yield a solution to (6) and (7) when the dynamic programming computation may be made. For example, one can use the following simple method adapted from one described by Dantzig[4] for the knapsack problem: Let $i_1, i_2, \cdots, i_m$ be such that $b_{i_1}/\ell_{i_1} \geqq b_{i_2}/\ell_{i_2} \geqq \cdots \geqq b_{i_m}/\ell_{i_m}$. Choose $a_{i_1} = [L/\ell_{i_1}]$, $a_{i_2} = [(L - \ell_{i_1} a_{i_1})/\ell_{i_2}]$, $a_{i_3} + [(L - \ell_1 a_{i_1} + \ell_{i_2} a_{i_2})/\ell_{i_3}]$, and so on. Only when this simple method has failed to provide a solution to (6) and (7) for all of the stock lengths would it be necessary to use dynamic programming to try to find a solution or show that no solution existed for (6) and (7) for any of the stock lengths.

In detail, a routine for determining a solution of (2) and (3) for which (1) is a minimum is the following:

(1) Determine m initial activities and their costs as follows: for each i, choose a stock length L_j for which $L_j > l_i$ and define the ith activity to be the one cutting $a_{ii} = [L_j/\ell_i]$ pieces of length ℓ_i from L_j. The cost of the ith activity will be the cost c_j of the stock length L_j from which the ith activity cuts the pieces of length ℓ_i.

(2) Form the $m+1 \times m+1$ matrix $\boldsymbol{B}$:

$$\begin{Vmatrix} 1 & -c_1 & -c_2 & & -c_m \\ 0 & a_{11} & 0 & & 0 \\ 0 & 0 & a_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & & \cdots \\ 0 & 0 & 0 & & a_{mm} \end{Vmatrix},$$

where a_{ij} is the number of pieces of length ℓ_i cut in the ith activity from a stock length whose cost is c_j. Record in some manner for each of the last m columns of $\boldsymbol{B}$ what activity corresponds to it. This record will be updated as improved solutions are found, and hence must also be able to indicate a correspondence between a column and a slack variable.

Form also $mm+1$ dimensional column vectors $\mathbf{S}_1, \cdots, \mathbf{S}_m$ corresponding to the slack variables, where the $\mathbf{S}_i$ has zeros everywhere except in the $(i+1)$st row where it has -1, and the $m+1$ dimensional column vector $\boldsymbol{N}'$ with 0 in the first row and $\mathbf{N}_i$ in the ith row and compute $\boldsymbol{B}^{-1}$ which is simply

$$\begin{Vmatrix} 1 & c_1/a_{11} & c_2/a_{22} & \cdots & c_m/a_{mm} \\ 0 & 1/a_{11} & 0 & \cdots & 0 \\ 0 & 0 & 1/a_{22} & \cdots & 0 \\ \cdots & & & \cdots & \\ 0 & 0 & 0 & \cdots & 1/a_{mm} \end{Vmatrix}.$$

Let $\boldsymbol{N} = \boldsymbol{B}^{-1} \cdot \boldsymbol{N}$. Given the current $\boldsymbol{B}^{-1}$ and the column vector $\boldsymbol{P}$ of a variable not used in the current solution (i.e., the first row of $\boldsymbol{P}$ is minus the cost, the other m rows are the coefficients a_{ij}; or in the case of the ith slack variable, $\boldsymbol{P}$ is S_i) to determine whether the current solution can be improved by making use of the variable it is only necessary to compute the first element of $\boldsymbol{B}^{-1} \cdot \boldsymbol{P}$; if it is not positive then no improvement will result, while if it is positive an improvement will result. Hence:

(3) The ith slack variable if it is not appearing in the current solution will lead to an improved solution if and only if the $(i+1)$st element of the first row of $\boldsymbol{B}^{-1}$ is negative.

(4) If no slack variable will improve the current solution it is necessary to determine whether the introduction of a new activity will improve the current solution by determining whether there is a stock length L with cost c for which inequalities (6) and (7) have a solution, where $b_1, \cdots, b_m$ are the last m elements in the first row of $\boldsymbol{B}^{-1}$. If these inequalities have no solution no matter which of the given stock lengths $L_1, \cdots, L_k$ with costs respectively $c_1, \cdots, c_k$ is tried, then the current solution is a minimum. The current solution and its cost is then given by the current $\boldsymbol{N}$, where the first row of this vector is the cost and the remaining m rows are, in

order, the values of the variables corresponding to the m columns of B^{-1}. If a new activity will improve the current solution form its column vector P with coefficients, in order, $-c, a_1 a_2, \cdots, a_m$.

(5) The introduction of either a slack variable or a new activity will improve the current solution. In either case let P be the column vector of the variable. To determine the new B^{-1} and N, which determine an improved solution and its cost, and which permit an iteration of these steps (3), (4), and (5) one proceeds as follows: Compute $B^{-1} \cdot P$; let the elements of this vector be $y_1, \cdots, y_m, y_{m+1}$ and let the elements of the current N be $x_1, \cdots, x_m, x_{m+1}$. Determine the i, $i \geqq 2$, for which $y_i > 0$, $x_i \geqq 0$, and x_i/y_i is minimum and let it be k; should the minimum ratio be zero one must use the method for degeneracy outlined in the next paragraph. If the minimum ratio is not zero then the kth element of P, y_k, will be the pivot element in a formal process of Gaussian elimination carried out simultaneously on B^{-1}, $B^{-1} \cdot P$ and N; that is elimination on the $(m+1) \times (m+3)$ matrix G formed from B^{-1} by adjoining as new columns N and $B^{-1} \cdot P$ in that order. (For each $i \neq k$ the kth row is multiplied by y_i/y_k and the resulting row subtracted from the ith; the kth row is simply multiplied by $1/y_k$.) The first $m+1$ columns of G' then form the new B^{-1} and the $m+2$nd column of G' forms the new N. The record of correspondence between columns of B^{-1} and activities or slack variables is updated by removing the current correspondent of the kth column and replacing it by the new activity or slack variable, whichever the case may be.

Degeneracy, should it occur, can be handled in much the usual way. Some precautionary device must be introduced to prevent the possibility of cycling. A new column N^1 of positive elements $x_1', \cdots, x_m', x_{m+1}'$ independent of N is adjoined to G and the choice of which $y_i > 0$ for which $x_i = 0$ is to be the pivot element is made on the basis of which such i has $x_i' > 0$ and x_i'/y_i minimum. When the pivot element has been chosen, Gaussian elimination proceeds as before but now on the enlarged matrix G. The additional column is maintained in G until an i exists for which x_i/y_i is positive and finite when the column can be dropped. Should it be the case that there is no i for which either x_i/y_i or x_i'/y_i is positively finite then another column N^2 of positive elements independent of N and N^1 must be introduced and used in the decision of the pivot element until such time as N or N^1 can be used. Similarly any number of columns N, N^1, N^2, $\cdots$ can be introduced as needed and dropped when no longer needed. Since the columns are independent when introduced and remain independent after Gaussian elimination, no more than m such columns need ever be introduced. Each column that is introduced defines a new linear programming problem that will prevent cycling as long as degeneracy does not occur within it.

EXAMPLE

THE MAIN steps in the computation of a solution of the following example are given below. An order for 20 pieces of length 2, 10 pieces of length 3 and 20 pieces of length 4 is to be cut from stock lengths 5, 6, and 9 with costs respectively of 6, 7, and 10.

Initially:

$$B=\begin{Vmatrix}1&-6&-6&-6\\0&2&0&0\\0&0&1&0\\0&0&0&1\end{Vmatrix},\qquad N'=\begin{Vmatrix}0\\20\\10\\20\end{Vmatrix},$$

$$B^{-1}=\begin{Vmatrix}1&3&6&6\\0&\frac{1}{2}&0&0\\0&0&1&0\\0&0&0&1\end{Vmatrix},\qquad N=B^{-1}\cdot N'=\begin{Vmatrix}240\\10\\10\\20\end{Vmatrix}.$$

The stock lengths will be tried in order of decreasing size since the longer stock length permits the definition of a larger number of activities. The first inequalities are therefore:

$$(1)\quad 9\geqq 2a_1+3a_2+4a_3 \text{ and}$$

$$(2)\qquad 3a_1+6a_2+6a_3>10.$$

The *ad hoc* method of solution gives (0, 3, 0) as a solution to (1) so that the new column vector P has elements $-10, 0, 3, 0$. Hence:

$$G=\begin{Vmatrix}1&3&6&6&240&8\\0&\frac{1}{2}&0&0&10&0\\0&0&1&0&10&\textcircled{3}\\0&0&0&1&20&0\end{Vmatrix},$$

where the last column is $B^{-1}\cdot P$, with the pivot element circled. After a Gaussian elimination on the pivot element

$$G'=\begin{Vmatrix}1&3&10\frac{2}{3}&6&640\frac{2}{3}&0\\0&\frac{1}{2}&0&0&10&0\\0&0&\frac{1}{3}&0&10\frac{2}{3}&1\\0&0&0&1&20&0\end{Vmatrix},\qquad B^{-1}\cdot P=\begin{Vmatrix}2\\\textcircled{2}\\0\\0\end{Vmatrix}.$$

Since none of the elements 3, $10\frac{2}{3}$, and 6 in the first row are negative, the introduction of a slack variable will not improve the cost. The inequality (2) is therefore replaced by $3a_1+10\frac{2}{3}a_2+6a_3>10$. The *ad hoc* method gives (4, 0, 0) as a solution so that $B^{-1}\cdot P$ is the vector listed next to G'

above with the pivot element circled. Thus,

$$G' = \begin{Vmatrix} 1 & 5/2 & 10/3 & 6 & 610/3 & 0 \\ 0 & 1/4 & 0 & 0 & 5 & 1 \\ 0 & 0 & 1/3 & 0 & 10/3 & 0 \\ 0 & 0 & 0 & 1 & 20 & 0 \end{Vmatrix}, \qquad B^{-1} \cdot P = \begin{Vmatrix} 2 \\ 0 \\ 0 \\ (2) \end{Vmatrix}.$$

Again the introduction of a slack variable will not improve the cost and therefore the inequality (2) is replaced by $5/2\, a_1 + 10/3\, a_2 + 6\, a_3 > 10$. The *ad hoc* method of solution gives (0, 0, 2) as a solution so that $B^{-1} \cdot P$ is the vector given next to G', with the pivot element circled. Thus,

$$G' = \begin{Vmatrix} 1 & 5/2 & 10/3 & 5 & 550/3 & 0 \\ 0 & 1/4 & 0 & 0 & 5 & 0 \\ 0 & 0 & 1/3 & 0 & 10/3 & 0 \\ 0 & 0 & 0 & 1/2 & 10 & 1 \end{Vmatrix}, \qquad B^{-1} \cdot P = \begin{Vmatrix} 1/2 \\ (5) \\ 0 \\ 0 \end{Vmatrix}.$$

Again the introduction of a slack variable will not improve the cost and therefore the inequality (2) s replaced by $5/2\, a_1 + 10/3\, a_2 + 5 a_3 > 10$. Now the *ad hoc* method of solution yields no solution to the inequalities. However, the *ad hoc* method can still be tried on the inequalities arising from the other stock lengths. The stock of length 6 gives rise to the inequalities:

$$(3)\ 6 \geqq 2a_1 + 3a_2 + 4a_3,$$

$$(4)\ 5/2\, a_1 + 10/3\, a_2 + 5\, a_3 > 7$$

for which the *ad hoc* method gives (3, 0, 0) as a solution. Therefore, $B^{-1} \cdot P$ is the vector next to G' above with the pivot element circled. Thus,

$$G' = \begin{Vmatrix} 1 & 7/3 & 10/3 & 5 & 540/3 & 0 \\ 0 & 1/3 & 0 & 0 & 20/3 & 1 \\ 0 & 0 & 1/3 & 0 & 10/3 & 0 \\ 0 & 0 & 0 & 1/2 & 10 & 0 \end{Vmatrix}, \qquad B^{-1} \cdot P = \begin{Vmatrix} 1/3 \\ (1/3) \\ 0 \\ 1/2 \end{Vmatrix},$$

and again since the introduction of a slack variable will not improve the cost, the inequality (4) is replaced by $7/3\, a_1 + 10/3\, a_2 + 5 a_3 > 7$. The *ad hoc* method yields (1, 0, 1) as a solution so that $B^{-1} \cdot P$ is the vector as is given above with the pivot element circled. Thus,

$$G' = \begin{Vmatrix} 1 & 2 & 10/3 & 5 & 520/3 & 0 \\ 0 & 1 & 0 & 0 & 20 & 1 \\ 0 & 0 & 1/3 & 0 & 10/3 & 0 \\ 0 & -1/2 & 0 & 1/2 & 0 & 0 \end{Vmatrix}, \qquad B^{-1} \cdot P = \begin{Vmatrix} 1/3 \\ 1 \\ (1/3) \\ 0 \end{Vmatrix},$$

and again since the introduction of a slack variable will not improve the cost the inequality (4) is replaced by $2a_1+1\frac{2}{3}a_2+5a_3>7$. The *ad hoc* method no longer yields a solution to the inequalities. Looking again at the other stock lengths, consider first inequality (1), together with:

$$(5)\quad 2a_1+1\tfrac{2}{3}a_2+5a_3>10.$$

The *ad hoc* method yields no solution. Consider next, the inequalities

$$(6)\quad 5\geqq 2a_1+3a_2+4a_3,$$

$$(7)\qquad 2a_1+1\tfrac{2}{3}a_2+5a_3>6,$$

which also have no solution by the *ad hoc* method. It is therefore necessary to use dynamic programming to maximize $2a_1+1\frac{2}{3}a_2+5a_3$ subject to (1), (3), or (6). Computing $F_3(x)$ we get $F_3(5)=5\frac{1}{3}$, $F_3(6)=7$, $F_3(9)=10\frac{1}{3}$. Since $5\frac{1}{3}\leqq 6$, $7\leqq 7$ the inequality pairs involving standard lengths 5 and 6 have no solution, but since $10\frac{1}{3}>10$ the inequalities (1) and (5) involving length 9 do have a solution (1, 1, 1), which has in fact been produced by our calculation. Therefore, $\boldsymbol{B}^{-1}\cdot\boldsymbol{P}$ is the vector listed above next to $\boldsymbol{G}'$ with the pivot element circled. Thus:

$$\boldsymbol{G}'=\begin{Vmatrix} 1 & 2 & 3 & 5 & 170 & 0 \\ 0 & 1 & -1 & 0 & 10 & 0 \\ 0 & 0 & 1 & 0 & 10 & 1 \\ 0 & -\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \end{Vmatrix},$$

and again since the introduction of a slack variable will not improve the cost the inequality (2) is replaced by: $2a_1+3a_2+5a_3>10$. The *ad hoc* method of solution yields no solution to these inequalities, nor does it yield a solution to the inequalities (3) and: $2a_1+3a_2+5a_3>7$, nor to the inequalities (6) and: $2a_1+3a_2+5a_3>6$.

It is therefore necessary to use dynamic programming again to maximize $2a_1+3a_2+5a_3$ subject to (1). This single computation yields $F_3(5)=5$, $F_3(6)=7$, $F_3(10)=10$, which shows that none of the pairs of inequalities are compatible. The solution is therefore to cut each of 10 pieces of stock length 6 into 1 piece of length 4 and 1 piece of length 2 and each of 10 pieces of the stock length 9 into 1 piece of length 2, 1 piece of length 3, and 1 piece of length 4. The cost is 170.

That integers should result as the solution of the example is, of course, fortuitous.

REFERENCES

1. Kurt Eisemann, "The Trim Problem," *Management Sci.* **3**, 279–284 (1957).
2. L. R. Ford, Jr. and D. R. Fulkerson, "A Suggested Computation for Maximal Multi-Commodity Network Flows," *Management Sci.* **5**, 97–101 (1958).

3. GEORGE B DANTZIG AND PHILIP WOLFE, "Decomposition Principle for Linear Programs," *Opns. Res.* **8**, 101–111 (1960).
4. ———, "Discrete Variable Extremum Problems," *Opns. Res.* 5, 161–310 (1957).
5. ALAN S. MANNE, "Programming of Economic Lot Sizes," *Management Sci.* **4**, 115–135 (1958).
6. A. CHARNES AND M. H. MILLER, "A Model for the Optimal Programming of Railway Freight Train Movements," *Management Sci.* **3**, 74–92 (1956).
7. R. E. GOMORY, "An Algorithm for Integer Solutions to Linear Programs," Princeton-IBM Mathematics Research Project Technical Report No. 1, November 17, 1958.
8. ———, "All-Integer Integer Programming Algorithm," IBM Research Report RC-189, January 29, 1960.
9. HARVEY M WAGNER AND THOMSON M. WHITIN, "A Dynamic Version of the Economic Lot Size Model," *Management Sci.* **5**, 89–96 (1958).