Integer Programming and Pricing
Author(s): Ralph E. Gomory and William J. Baumol
Source: *Econometrica*, Vol. 28, No. 3 (Jul., 1960), pp. 521-550
Published by: The Econometric Society
Stable URL: http://www.jstor.org/stable/1910130
Accessed: 13/09/2010 13:37

Econometrica, Vol. 28, 3 (July 1960)

INTEGER PROGRAMMING AND PRICING

By Ralph E. Gomory and William J. Baumol

In this article Gomory's method of solution of integer linear programming problems is described briefly (with an example of the method of solution). The bulk of the paper is devoted to a discussion of the dual prices and their relationship to the marginal yields of scarce indivisible resources and their efficient allocation.

1. INTRODUCTION

It has been known for some time that a method of solution of the general linear programming problem in which the variables are required to take integer values would also permit the solution of a considerable variety of other problems many of which are not obviously related to it.[1] For example, Markowitz and Manne [**13**] have shown that the difficult concave (nonlinear) programming problem (e.g., a cost minimization problem in which the total cost function is shaped like a hill) can, at least in principle, be approximated as an integer program which permits the determination of a global, and not just a local minimum. Nonconvex feasible regions can also, at least in principle, be handled by integer programming. Among the economic problems which are related to integer programming are the travelling salesman problem and problems in which fixed (inescapable) costs are present. A surprisingly wide range of problems including diophantine problems and the four color map problem[2] can be given an integer programming formulation. Some of these applications will be described in greater detail in section five of this paper.

Recently one of the authors of this article developed a method, which he calls the method of integer forms (MIF), for solving integer programming problems. In the next section the method of solution will be described in some detail. No proof that the algorithm arrives at the optimal integer solution in a finite number of steps will be described since it is rather lengthy and is being published elsewhere (see Gomory [**6**] and [**7**]. For an alternative approach see Land and Doig [**12**]).

The bulk of the paper, however, will be devoted to a discussion of the pricing problem in the integer programming case, that is, in the case where,

[1] For an excellent survey of the applications of integer programming see Dantzig [**3**].

[2] Integer programming methods have not succeeded either in confirming or rejecting such conjectures as the four color hypothesis. Rather, the technique permits the solution of individual problems *when solutions exist*. Thus, for any specific map, *if there exists a solution* to the four color map problem, integer programming can be used to find a solution, i.e., to assign four colors among the different territories in such a way that no two territories with the same color have a common boundary.

in economic terms, the inputs and outputs are "lumpy" (not perfectly divisible). We shall show that the MIF algorithm produces a dual problem whose solution also imputes shadow prices to the scarce inputs. These prices possess a number of the properties of the dual prices of ordinary linear programming. In particular, they possess one of the most important properties of ordinary dual prices—*they permit the construction of a decentralized decision making arrangement which, in principle, will achieve some of the possible efficient allocations of resources*. We shall see, however, that the price system no longer suffices to achieve *every* efficient allocation, and that, when the consumer side of the market is taken into account, the entire ideal output theorem of perfect competition runs into difficulties. These integer dual prices also possess a number of peculiar features. First, they will themselves be integers. Second, they are to some extent arbitrary and will vary with the procedure by which they are computed. Third, they will tend to impute a zero price to a number of resources to which the economist will want to assign a higher value. Fourth, the dual price of a resource will not always be equal to its marginal revenue product, and, in fact, the marginal revenue product of an input itself becomes a somewhat ambiguous concept.

Before actually describing the method of integer forms it seems worthwhile to state the result it produces, a result very similar to the one produced by the ordinary simplex method.

In the ordinary simplex method, starting with the integer inequalities in n original variables x_j

$$\sum_{j=1}^{n} a^*_{i,j}x_j \leq Q_i \qquad (i = 1, \ldots, m,\ a^*_{i,j} \text{ integers}) \tag{1.1}$$

and an objective function

$$z = a^*_{0,0} + \sum_{j=1}^{n} a^*_{0,j}\,(-x_j) \qquad (a^*_{0,j} \text{ integers}),$$

one introduces slack variables x'_i, one for each inequality, converting them into equations

$$x'_i = a^*_{i,0} + \sum_{j=1}^{n} a^*_{i,j}\,(-x_j) \qquad (i = 1, \ldots, m)\,, \tag{1.2}$$

where the $a^*_{i,0}$ are the Q_i of (1.1).

In the usual language of linear programming the x'_i in (1.2) are the "basic" variables, the x_j are the "non-basic" ones. In the simplex method one tries out in succession different sets of basic and non-basic variables, each time changing one variable from "basic" to "non-basic" and vice versa. Every such interchange of two variables is referred to as a "pivot step" of the simplex method. After a series of such steps (1.2) becomes (1.3)

$$(1.3)\qquad \begin{cases} z = a_{o,o} + \sum_{j=1}^{n} a_{o,j}(-t_j)\,, \\ t'_i = a_{i,o} + \sum_{j=1}^{n} a_{i,j}\,(-t_j) \qquad (i = 1, \ldots, m)\,, \end{cases}$$

where the t'_i are the current basic variables and the $a_{i,j}$ are the coefficients used to express them in terms of the current non-basic ones, the t_j. The simplex method guarantees that eventually we can reach an expression (1.3) in which all the $a_{i,o}$, $i = 1, \ldots, m$, are nonnegative and also all the $a_{o,j}$, $j = 1, \ldots, n$, are nonnegative.

At this point we have obtained an optimal solution, for in order to maximize z all the non-basic variables (the t_j) must then be set equal to zero since every nonzero t_j must involve some subtraction from z (first equation in (1.3)). Each basic variable t'_i must now be equal to the appropriate $a_{i,o}$ since all other terms in the equations drop out. This, then, is the solution to the programming problem where it will be noted that, since all $a_{i,o}$ are nonnegative, all variables automatically get nonnegative values, as required. The $a_{o,j}$ also have economic significance, for, as will be noted later in this paper, they are the shadow prices of the dual problem.

In the method of integer forms one proceeds exactly as in the simplex method, only from time to time certain new variables and inequalities, which will be described presently, are added to the problem. The result is again a final set of equations

$$(1.4)\qquad \begin{cases} z = a_{o,o} + \sum_{j=1}^{n} a_{o,j}(-t_j)\,, \\ t'_i = a_{i,o} + \sum_{j=1}^{n} a_{i,j}(-t_j) \qquad (i = 1, \ldots, m')\,, \end{cases}$$

with the $a_{i,o}$ and $a_{o,j}$ (except possibly $a_{o,o}$) nonnegative, and again the solution to the linear programming problem is obtained by setting $t'_i = a_{i,o}$. But this time there is a different number of equations, m', (where $m \leq m' \leq m + n$). These involve m' basic variables t'_i all of which were present in the original equations. There are still only n non-basic variables, t_j. However, while some of these are variables may appear in the original equations, others may be new variables added during the course of the computation. The essential point is that now *all the $a_{i,j}$ are integers*. Thus the solution is in integers.

2. THE METHOD OF SOLUTION

A geometric picture of the integer programming problem will give the reader an intuitive grasp of the method of solution. In Figure 1 we represent the feasible region, $OABCD$, of an ordinary linear programming problem. The dots within this region represent all feasible points both of

whose coordinates are integers (the integer lattice points). The solution to the ordinary programming problem will occur on the boundary of the feasible region and in the diagram none of the boundary (other than the origin) goes through an integer lattice point. Suppose, however, that the feasible

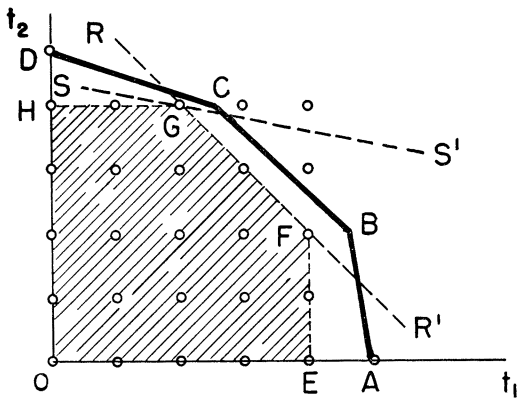

FIGURE 1

region could somehow be shrunk to the convex hull of the feasible lattice points (the shaded region). It is to be noted that this, too, would be the diagram of a linear programming problem—in fact, of the original problem modified by the addition of several supplementary linear constraints such as RR'. This new programming problem has two other important features: (1) it includes every integer feasible solution to the original program and (2) every basic (corner) solution of the new problem is an integer solution, for the boundary of the convex hull consists of linear segments which join lattice points. It follows at once that a (basic) optimal solution to the new programming problem must be an optimal integer solution to the original problem.

In practice it is difficult to cut the feasible region down to the convex hull of the feasible integer lattice points. The MIF method of solution does consist of a sequence of steps involving the addition of constraints to the original linear program and the subsequent solution of this expanded linear program. These constraints are chosen in a way which gives them the following properties: (1) they normally reduce the feasible region; (2) their graph (e.g., SS') ordinarily goes through at least one lattice point (point G in the figure) but it must be emphasized that *this lattice point need not lie in the feasible region*; (3) they never exclude from the new feasible region a lattice point which was originally feasible; and (4) they produce, in a finite number of steps, a new linear program whose solution is in integers and which is therefore the optimal integer solution of the original programming

problem (if any such solution exists). It is to be noted that the feasible region of this final programming problem will include $OEFGH$, the convex hull of the feasible lattice points, and will itself be included in the original feasible region, $OABCD$. In this diagram it is clear that, with a suitable objective function, an optimal integer solution will occur, e.g., at point G at the intersection of the additional constraint lines RR' and SS'.

It remains now to describe the construction of these additional constraints. Consider any equation that occurs in the course of the solution of the problem which we write as

$$t_i' = a_{i,o} + \sum_{j=1}^{n} a_{i,j}(-t_j) \qquad (i = 0, \ldots, n) \tag{2.1}$$

in which we happen to have $a_{i,j} \geq 0$ for $j = 0, \ldots, n$. (Slack variables have been included in the set of variables t where necessary so that each constraint is an equation.) For later reference note that (2.1) *may be the objective (profit) function* $z = a_{o,o} + \Sigma a_{o,j}\,(-t_j)$, where z is the variable whose value is to be maximized (total profit). In other words, (2.1) need not be a constraint equation. For reasons which will later be emphasized, the t_i' are the current basic variables and the t_j are the current nonbasic variables.

Write $a_{i,o} = k + f_{i,o}$ where k is an integer and $0 \leq f_{i,o} < 1$. Inserting this in (2.1) and rearranging we have

$$\Sigma a_{i,j}t_j = f_{i,o} + k - t_i' \,. \tag{2.1a}$$

Since the $a_{i,j}$ are assumed ≥ 0, and the t_j are required to be nonnegative, the left hand side in (2.1a) must be a *nonnegative* number. The left hand side, however, differs from $f_{i,o}$ only by the integer $k - t_i'$. So the left side can only be one of the nonnegative numbers $f_{i,o}$, $1 + f_{i,o}$, $2 + f_{i,o}$, . . ., etc.

In every case

$$\Sigma a_{i,j}t_j \geq f_{i,o} \tag{2.2}$$

an additional inequality which must clearly be satisfied by any nonnegative integer solution to our original problem.

Further, since we require the t_j to be integers, any change in the $a_{i,j}$ by an integer amount to, say, $a_{i,j}^*$, where $a_{i,j}^*$ is also nonnegative, produces another valid inequality such as(2.2). For such a change in $a_{i,j}$ must change $a_{i,j}t_j$ and hence $\Sigma a_{i,j}t_j$ by an integer amount, so that $a_{i,o} + \Sigma a_{i,j}^*\,(-t_j)$ must still be an integer, call it t_i^*. Repeating our reasoning with t_i^* replacing t_i' and the $a_{i,j}^*$ replacing the $a_{i,j}$ we obtain

$$f_{i,o} \leq \Sigma_j a_{i,j}^* t_j \,, \qquad a_{i,j}^* \geq 0 \,. \tag{2.3}$$

The strongest possible inequality which can be obtained by this process is clearly

$$f_{i,o} \leq \sum_{j=1}^{n} f_{i,j} t_j , \qquad \text{all } f_{i,j} \geq 0 , \tag{2.4}$$

where the $f_{i,j}$ are the fractional parts of the $a_{i,j}$ in (2.2).

Note further that while in (2.1) we assumed $a_{i,j} \geq 0$ this restriction is unnecessary for the derivation of inequalities (2.4). For if in some original constraint we have, say, $a_{i,k} < 0$, since t_k is required to be an integer, we can *increase* the $a_{i,k}$ by some integral amount to obtain $a^*_{i,k} \geq 0$ and so, repeating this procedure for all other negative coefficients, we end up with an inequality of form (2.3) from which we can again obtain (2.4). These inequalities (2.4) or the corresponding equations

$$s_i = -f_{i,o} - \Sigma f_{i,j}(-t_j) , \qquad s_i \geq 0, \text{ all } f_{i,j} \geq 0 , \tag{2.5}$$

where s_i is a slack variable, are the additional restrictions (corresponding to the equations of lines such as SS' in Figure 1) that are employed in solving the integer programming problem. Other restrictions can be obtained by adding together two or more equations or integer multiples of equations and then deducing a new restriction from the new combined equation. The class of possible restrictions is discussed in [**6**] where these restrictions are shown to form a finite group under certain simple rules of combination.

Several characteristics of (2.5) are to be noted: (1) s_i is itself required to be an integer for it is the of k—t' term of (2.1a); (2) if the optimal solution of the original programming problem contains any noninteger values it will not satisfy (2.5) so that (2.5) normally excludes some of the original feasible region;[3] (3) by the nature of its construction any feasible integer solution of the original programming problem will satisfy (2.5) so none of the original feasible lattice points is excluded by (2.5); finally, (2.4) is usually satisfied as an equality by some (not necessarily feasible) lattice point (so that here $s_i = 0$ in (2.5)).[4]

[3] For whenever the constants, $a_{i,o}$, in (2.1) are nonnegative, the solution obtained by setting all the non-basic t_j equal to zero is a feasible one. However, setting all t_j equal to zero in (2.4) violates that inequality except in the case $f_{i,o} = 0$. If any solution is noninteger so that $f_{i,o} \neq 0$, the formerly feasible point in which $t_j = 0$, all j, is thus excluded by the new inequality (2.4).

This also shows that constraint (2.5) normally cuts off some of the "top" of the feasible region in Figure 1 (SS' cuts out optimal point C) despite the direction of the inequality in (2.4) which seems to make it cut off a bottom piece. The explanation is that Figure 1 and (2.4) are expressed in terms of different variables. The t's in the figure are all in the current basis and hence, usually, positive while the variables on the R.H.S. of (2.4) are all initially nonbasic so that no constraint can decrease their (zero) values any further.

[4] For suppose all the f's in (2.5) are rational. Then that equation may be rewritten as $s'_i = F_{i,o} - \Sigma F_{i,j}(-t_j)$ where all the $F_{i,j}$ are integers. The greatest common

The algorithm for solving an integer programming problem is then: *step* 1, solve the original problem; *step* 2, if the solution is noninteger add any additional constraint (2.5); *step* 3, repeat this process until an integer solution (if any exists) is obtained. It should be noted that some of the additional constraints will become redundant and can be dropped so that no more than n additional constraints will ever be required at any one time.

It is also important from the practical point of view to realize that the sucessive reoptimizations usually require only a few steps.[5]

It will be noted that at any intermediate stage of this process there will usually be a number of possible constraints of the form (2.5). Any one of them can be used.[6] The solution process may be hastened, however, if in some sense the inequality (2.5) is chosen so as to make some sort of average $f_{i,o}/f_{i,j}$ as large as possible. The reason for this is most easily seen geometrically. In terms of Figure 1, our objective is to choose (2.5) is such a way that its associated graph SS' cuts off as much as possible of the "redundant" feasible region (the unshaded portion of the original feasible region). But at the old optimal point, C, all of the t_j in (2.5) were zero, i.e., this constraint is expressed in terms of the old *non*-basic variables. In other words, to move as far as possible from point C we require these formerly zero t_j to be *increased* as much as they can be. That is, we wish the hyperplane in these t_j obtained by setting $s_i = 0$ in equation (2.5) to be as far from the origin in their subspace as is possible. But $f_{i,o}/f_{i,j}$ is the nonzero coordinate of this plane on the t_j axis, so that by making these fractions as large as possible we bring this plane as far as we can from the origin.

In the illustrative computation below we shall employ only the roughest

---

divisor, G, of the $F_{i,j}$ $i \neq 0$, can be represented as an integer combination of these numbers so that we have $G = -\sum F_{i,j}(-t'_j)$ with the t'_j integers. If the $F_{i,j}$ have no common divisor we obtain $G = 1$ so that multiplying through by the integer $F_{i,o}$ and writing $t^* = t'F_{i,o}$ we have the integer solution $F'_{i,o} = -\sum F_{i,j}(-t^*_j)$ for which s'_i and hence $s_i = 0$.

[5] This is because the problem, before the additional constraint has been added, has been brought into optimal form, i.e., it is both primal feasible (all $a_{i,o} \geq 0$, $i \neq 0$) and dual feasible (all $a_{o,j} \geq 0$, $j \neq 0$) (see Section 6, below). After the constraint is added it is still dual feasible, and only one $a_{i,o}$, the $-f_{i,o}$, added is negative, consequently, using the dual simplex method, the problem can usually be brought back to optimal form quite rapidly.

[6] The proof that the process terminates in a finite number of steps given in [6] actually requires that new inequalities be chosen by a certain rule. The proof can easily be extended so that the rule need be followed only once every p steps, p a fixed integer, and a free choice made the rest of the time. In actual computations so far what has been done was to choose a large f_o as described above. Some recent computations indicate, however, that the finiteness rule may have to be followed on larger problems.

approximation to this ideal by choosing that inequality (2.5) for which $f_{i,o}$ is as large as possible.

3. THE DUAL PRICES AND MARGINAL VALUATION

The solution to the integer programming problem which has just been described involves the solution to an ordinary linear programming problem which is identical with the original program except for the addition of several (at most n) "artificial" constraints. For convenience we may refer to this new program as the *augmented linear program.* Clearly, as to any linear program, there is a dual program which corresponds to this augmented program. Moreover, if the augmented program has a solution, i.e., if the original program has any integer solution, the dual problem, too, will have a solution which consists of the shadow prices corresponding to the constraints of the primal problem (where the primal problem is interpreted as that of selecting the optimal levels of several activities).

These dual prices are obtained just as they are in linear programming. If in the solution to an ordinary linear programming problem, the t_j in (1.3) is the slack of the kth constraint (the one involving the kth good), then the $a_{o,j}$ for that j is the shadow price of the kth good. In (1.4) the t_j may be slacks of original constraints or added ones, but the prices are determined in just the same way. Since in (1.4) *all* of the $a_{i,j}$ are integers, the prices will be integers.

Since these prices are the solution to an ordinary linear programming problem they will possess the usual characteristics of ordinary dual prices. They will be nonnegative; except in cases of degeneracy they will impute zero profits to any activity that is carried on at a nonzero level in an optimal solution and negative profits to all other activities; they will make the total imputed value of all "scarce inputs" equal to the value of the optimum output combination; zero prices will be imputed to inputs that are not used to capacity, etc.

In several respects, however, these integer programming prices will be peculiar. As just indicated, the prices will themselves be integers. More important, these prices will vary with the choice of additional constraints (2.5). Finally, we note that prices will be imputed not just to the scarce facilities of the original program: corresponding to each of the added constraints of the augmented program there will also be a shadow price. Before discussing the prices corresponding to these added constraints (call them the artifical capacity prices) let us see what happens to the prices of the original scarce facilities.

Some of these prices may have risen. For example, in Figure 2 suppose C is the optimal solution to the noninteger program, that T is the optimal integer solution and that SS' is the added constraint of the augmented

program. Then the input associated with constraint AA' is not used to capacity at C but it is at T. Hence its price will be zero in the noninteger program and rise to some positive value in the integer program.

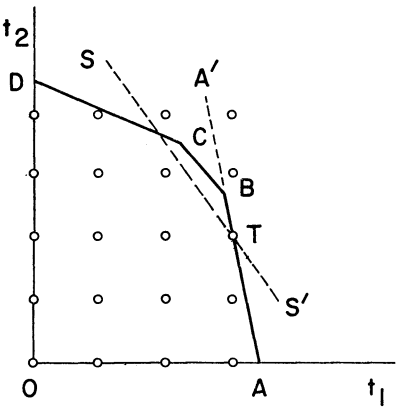

FIGURE 2

Many prices, however, which would be positive in a noninteger solution may be expected to fall to zero in the integer programming case. Thus the prices corresponding to constraint segments DC and CB are both positive when C is optimal, but they are both zero at the integer optimum T. The economic interpretation of these zero prices is easily given. If a warehouse has a capacity to store 36,463.4 cases of some item, an integer solution requires that the last 0.4 storage capacity be left empty—cases just do not come in fractional batches. But the calculation takes some of the warehouse's space to be idle, and hence labels it a free good—it is given a zero price.[7] Clearly this is not a fully satisfactory imputed price. We will return to this issue presently.

We see then that the requirement that the solution be in integers may increase some shadow prices and will normally reduce others. However, if we

[7] It is tempting to jump to the conclusion that "almost all" prices of original facilities will be driven to zero, for if, e.g., there is one capacity C, and one output each unit of which uses up X units of capacity, then between any two adjacent integer values of C/X, say n and $n + 1$, there will be a nondenumerable infinity of noninteger values of C/X for which it will be impossible to use up the capacity completely. In practice, however, this observation seems to be an exaggeration. Experience in problem solving shows that nonzero dual prices occur frequently. We seem to make up problems in a way which leads to this occurring. The same phenomenon is encountered elsewhere, say in the solution of linear difference equations where unit or multiple roots occur with a frequency which is surprising in view of the fact that the equations which possess such roots constitute a subset of measure zero of the set of all possible linear difference equations.

know that the capacities, Q_i, of the scarce facilities, including the capacities associated with the additional inequalities, are all nonnegative (as is proved in Appendix A) it is easy to show that the *arithmetic mean price* of the original facilities (when each price is weighted by the capacity of the corresponding facility) must fall. For let the augmented program have m original constraints and n additional constraints. Let the optimal noninteger prices be $P_1, \ldots, P_m$ and let the optimal integer prices be $P_1^*, \ldots, P_{m+n}^*$. Finally let the capicities of the scare facilities be $Q_1, \ldots, Q_{m+n}$. Then, since the additional constraints can never increase the maximum profit from the total output (equals the total imputed value of the scarce facilities), we have $\sum_{i=1}^{m} P_i Q_i \geq \sum_{i=1}^{m+n} P_i^* Q_i \geq \sum_{i=1}^{m} P_i^* Q_i$. Dividing through by $\sum_{i=1}^{m} Q_i$ we obtain the desired result.

There is a lower bound to this fall in average price. For suppose of the various constraints that could have been added in the augmented linear program we had chosen those which correspond to the boundary $EFGH$ of the convex hull of the lattice points (Figure 1). Since no constraint line SS' of our original augmented program has any points interior to this convex hull it can be added to the convex hull augmented program without affecting its solution. It follows that the convex hull augmented program consists of any other augmented program *plus some additional constraints*. It is then a direct consequence of the preceding theorem on average prices that the average dual price of the original capacities in any other augmented program will be greater than or equal to that of the convex hull augmented program. It is tempting to consider the latter to be the "true" integer programming prices since the convex hull of the integer lattice points represents the smallest convex body containing the entire integer feasible region (it can be shown though that even these prices may themselves not be uniquely determined. This is because what would be called degeneracy in ordinary linear programming is particularly likely to arise in integer problems). We would then say that the computed dual prices are usually overvalutions of the "true" dual prices. However, it will be shown, presently, that any such prices are themselves likely to be undervaluations of the marginal value product of a capacity.

So much for the prices of the original facilities. There remains the problem of interpreting the prices which correspond to the addition constraints (2.5). These may be viewed as a measure of the opportunity cost of indivisibility—e.g., the loss imposed on the businessman by a unit of the artificial capacity constraint which prevents him from seeking to stuff that last four tenths of a case into his warehouse. This interpretation, however, amounts to our thinking of these prices as the marginal revenue products of these inputs and we shall see now that, in the integer programming case, this concept runs into difficulties.

The basic difficulty involved in evaluating marginal revenue products in integer programming is that inputs come in indivisible units. For that reason we cannot speak, e.g., of the marginal profit contribution of a small change in input, i.e. we must deal with $\Delta R/\Delta X$ rather than dR/dX where ΔX is an indivisible unit of input X and R is total profit. But the dual prices represent dR/dX which may change over the range of a unit change in X.

More specifically, in Figure 3, we consider the effect of a unit decrease in the capacity, X, of the facility associated with constraint line UU where

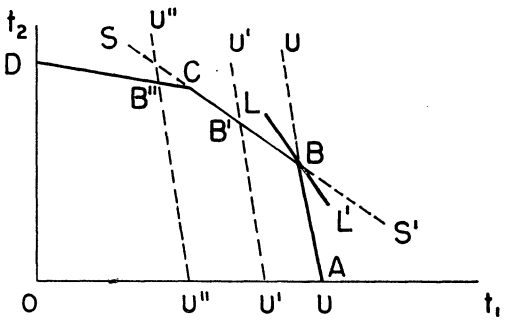

FIGURE 3

B is the original optimal point. Suppose this produces a shift to constraint line $U'U'$ which intersects CB at B'. This means that the new corner B' still lies on the intersection of the same constraint lines as before. The decrease in profits produced by the shift is strictly proportionate to the distance of the shift because the iso-profit lines such as LL' are all parallel straight lines. Hence dR/dX is constant over this range and equal to $\Delta R/\Delta X$ so that the dual price is equal to the marginal profit contribution of X as it would be in an ordinary linear programming problem.

But if a unit decrease in X shifts UU beyond $U'U'$ to $U''U''$ (past a corner, C, of the original feasible region) it is clear that dR/dX will change (more specifically, its absolute value will increase) beyond point C. In that case the marginal profit contribution of X, that is, $\Delta R/\Delta X$ will no longer be equal to dR/dX at the optimal point, B, which is the value of the computed dual price of X.

This argument also indicates, incidentally, why the value of the computed dual price will vary with the choice of additional constraint (2.1). Thus let SS' be the graph of such a constraint. Note that there is considerable choice in the slope of such a line, for so long as it goes through point B and has a negative slope less than that of the iso-profit line LL' it will still lead to the same optimal (integer) point B. But the value of the dual price of X, dR/dX at B, varies with the slope of SS' as we have just seen.

Normally, then, in integer programming there will be *three* marginal revenue product figures: dR/dX, $\Delta R/\Delta X^-$ where ΔX^- is a unit *decrease* in X, and $\Delta R/\Delta X^+$ where ΔX^+ is a unit *increase* in X.

For reasons which have just been indicated we shall normally have (in absolute value) $\Delta R/\Delta X^- \geq dR/dX$ (the dual price), that is, a unit decrease in X will reduce the objective function by no less than its dual price. It can be shown by numerical examples, however, that $\Delta R/\Delta X^+$ may be either greater or smaller than dR/dX. The reason is that an outward shift in one of the constraints can change the shape of the feasible region in a fairly unpredictable manner, because the change in this constraint can in turn cause a shift in some of the artificial constraints which are derived from it. This phenomenon does not affect the value of $\Delta R/\Delta X^-$ because when the feasible region is *reduced* any constraint which was initially valid will still be valid since its graph cannot lie inside the smaller feasible region.

There is one last matter to be discussed in this section. As mentioned before, the prices we have obtained have the unsatisfactory feature that they give zero prices to goods not normally considered free goods, goods that would be useful if available in larger quantities. The positive prices tend to be awarded instead to new "artificial goods" (capacities) whose limited availability shows up in the new inequalities. However, as a generalization of equation (A.1) of Appendix A to the n artificial constraint case shows quite clearly, the new inequalities are merely weighted sums (with nonnegative weights) of the old inequalities where we may use the symbols $g_{i,j}$ to designate the weight which is given the old inequality, j, in the expression for any new inequality, i. This suggests that the prices associated with the new inequalities might well be imputed or distributed back to the original goods (including some of those with zero prices) whose limited availability lies behind the scarcity of the artificial goods.

Appendix B is an attempt in this direction. The method proposed there can be described as follows. Let π_i represent the price of any artificial good i. In imputing back, we then add to the price, π_j, of any initial input good (capacity) j the amount $g_{i,j}\pi_i$. In other words, we obtain the recomputed prices

$$\pi'_j = \pi_j + \sum_i g_{i,j}\pi_i\,, \qquad \text{all } g_{i,j} \geq 0\,,$$

$$\pi'_i = 0\,,$$

for all artificial constraints i, where we note, incidentally, that we do not normally have

$$\sum_j \pi'_j = \sum_j \pi_j + \sum_i \pi_i\,.$$

These recomputed prices have the following desirable properties in common with ordinary linear programming dual prices, as is shown in Appendix B:

1. These prices are sufficiently high to eliminate the possibility of any profitable output, and an output will be produced if and only if it yields zero profits.

2. Any input with a zero recomputed price will be a free good in the true economic sense. That is to say, an unlimited increase in the stocks of this good will make absolutely no difference to optimum output levels.

Although the recomputed prices depend on the actual course of the calculation, as is shown in Appendix B, there is one case in which a type of uniqueness prevails.

3. If there is some set of n original inequalities such that these n alone determine the same integer solution as does the full set of inequalities, then, if all other inequalities are dropped, the recomputed prices for the reduced problem are unique, and are identical with the prices obtained by solving the reduced problem as an ordinary noninteger linear programming problem.

Aside from this, virtually nothing is known about the possible range of recomputed dual prices and the interpretation of this range.

The recomputed prices, however, will also have a number of unusual characteristics:

1. The converse of the preceding proposition 2 does not hold; that is, some free goods may not be given zero recomputed prices. This is because more than one subset of the constraint set may suffice to produce the ultimate optimal integer solution. In that case any one constraint which is not common to all such subsets can be considered redundant (i.e., to represent a free input) since elimination of that one constraint will make no difference to output levels. But it is not possible to eliminate all such constraints and so at least some of these must be chosen to receive a nonzero price. It should be noted that a similar situation can arise in an ordinary linear programming problem in cases of degeneracy.

2. Among the inequalities which make up the artificial constraints there may be included some of the final output nonnegativity conditions, $x_i \geq 0$. It follows that *some of the artificial constraint prices may be reimputed, in part, to some of these final outputs*. In other words, the process of price recomputation may well result in some changes in the prices of final outputs (activities) from the values given by the coefficients of the objective function. For purposes of the next section such a price change may conveniently be visualized as a per unit subsidy to the final outputs or activities affected.

4. PRICING, RESOURCE ALLOCATION, AND COMPETITION

Let us now see what role integer dual prices can play in welfare economics, and, in particular, in an arrangement for achieving an optimal allocation of resources through decentralized decision making.

Let us first note that the prices of the artificial constraints of the integer programming problem can be made very real to a firm by debiting them for the use of these artificial scarce resources. That is, if the fifth additional constraint involves the term $3t_7$, and the corresponding price is set at twelve dollars per unit, the firm would on this arrangement be charged 36 dollars for the use of this "scare resource" for every unit of output 7 it produced. Alternatively. the same resource allocation effect could be achieved by the use of "imputed back" prices as described in the previous section.

Suppose then that either a competitive market or a central planning authority were to compute the dual prices and output combinations necessary to maximize the value of total final output at any *fixed* set of commodity prices. It will be recalled that any such output combination must be an efficient output.[8] Moreover, if individual firms are charged for the use of both real and artificial scarce resources either directly at the computed dual prices or indirectly at these prices as imputed back to the original scarce resources, they will be forced to produce only the outputs contained in this efficient bundle since, by the usual properties of dual prices, each unit of any other output will incur a loss. If these outputs are then expanded as far as possible it follows that the firms must end up producing the efficient output in question.

We see then that every value maximizing (competitive) output will, by the usual argument, also be efficient, even in the integer programming case.[9] Unfortunately, the converse does not hold. There may be efficient outputs which are not competitive,[10] i.e., for which there exist no prices, P_i, at which this output combination maximizes the total value of output, $\Sigma P_i t_i$. This is easily proved by counterexample, as shown in Figure 4. Here the shaded triangle, OBC, is the convex hull of the feasible lattice points. Point A, with coordinates (2.1), lies in the interior of this triangle. But (because the feasible points are isolated) it is possible for such an interior point to be efficient. This is in fact the case with A for there is no feasible lattice point which "dominates" A, i.e., no point which lies directly above it, directly to the right of it, or above it and to its right. Now consider any straight line, such as PP' (equation $\Sigma P_i t_i = k$), through A. Any such

[8] For if the output combination Q were not efficient then there must, by definition, be some other output combination, Q', which contains larger outputs of some items and no smaller output of any item. Hence at the fixed prices the value of Q' must exceed that of Q, i.e., if Q is not efficient it cannot maximize the value of output.

[9] For the classic discussions of the problems of this section see Koopmans [**8**, Chapter 3], Arrow [**1**], and Debreu [**4**]. See also Dorfman, Samuelson and Solow [**5**, Chapter 14] and Koopmans [**9**, Essay 1].

[10] This has already been suggested by Koopmans and Beckman [**10**].

line must lie below either lattice point B or lattice point C. This means that there must exist another parallel line such as $P''P'''$ ($\Sigma P_i t_i = k^* > k$), which lies above PP' and goes through one of these corners of the convex hull triangle OBC. In other words, in the case in Figure 4, at the

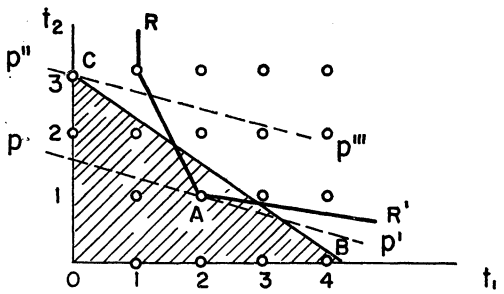

FIGURE 4

prices involved in the price (iso-output-value) lines shown, the value of output at point C exceeds that at A. And, similarly, at any other possible set of output prices the value of output at A will be smaller than that at B or that at C. This shows how there are, in the discrete programming case, likely to arise efficient outputs which are not competitive outputs and which cannot be enforced by the standard type of decentralized control procedure of the economic literature, in which the central authority makes only simple price decisions.

It is to be noted, however, that it is possible to find families of nonlinear or piecewise linear price curves such as RR' for which the value of output is maximized at A. This has a simple interpretation. The prices which are set up are discriminatory and vary with the magnitude of output. Output combinations which are close to A are given relatively high prices but as outputs move further and further from A prices are made increasingly unfavorable to the seller so that there are sharply diminishing returns to departures from A. In other words, an output, t_1, of any commodity at A is broken arbitrarily into a sum of sub-outputs $t_{11} + t_{12} + \ldots + t_{1n} = t_1$ and each of the sub-outputs t_{1i} is assigned a different price, P_{1i} as just described. Such an arrangement could, in principle, be enforced by government fiat. But it is difficult to see much advantage to a decentralized control procedure when it becomes so complicated, and in any event it would never result from the spontaneous operation of competitive market forces which preclude the existence of different prices for different units of a homogeneous product.

The so-called basic theorem of welfare economics runs into even more serious trouble in integer programming. It is in this situation not generally

possible to attain a Pareto optimal point by means of a price system.

This is obviously so for the case of interior efficient points such as A in Figure 4. For let RR' now represent a community indifference curve so that A is now the optimal feasible point. There obviously exists no line that separates the remainder of the feasible lattice points from the region socially preferred to or indifferent with A (the region above RR'). This means that with any fixed price arrangement producers will find it more profitable to manufacture either output combination B or C than to turn out the social optimum combination, A.

Moreover, even if the optimum point Q is a corner of the convex hull of feasible lattice points there may well exist no hyperplane which separates the feasible (producible) points from the lattice points which are preferred to or indifferent with Q. A way in which this may arise is illustrated in the following three dimensional diagram (Figure 5). Here the shaded region is

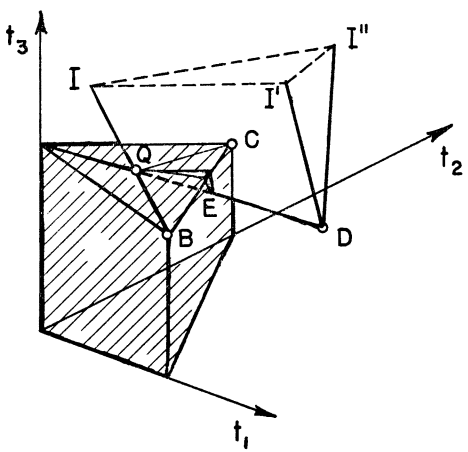

FIGURE 5

the convex hull of the producible (feasible) lattice points such as B, C and the optimal point Q. Similarly let $II'I''$represent a portion of the convex hull of the lattice points which are preferred to or indifferent with Q. It will be noted that Q is indifferent with nonfeasible lattice point D. The segment QE of the line QD, which connects indifferent points Q and D, lies below triangle QBC which forms one of the faces of the convex hull of the feasible region. Q is the (unique) optimal point because no other feasible lattice point lies (on or) above "indifference surface" $II'I''$. It is obvious that no plane surface can separate the feasible lattice points in the figure from the lattice points preferred to or indifferent with Q since any such plane must either lie above point D which is indifferent with Q or it must be below feasible points B and C.

Let us summarize the results of this section:

1. Every competitive output combination is efficient and any such point can be attained by a system of fixed prices set by central authority, all other decisions being left to the individual firms in the economy. This is no different from the result for the ordinary linear programming case.

2. Unlike the ordinary linear programming case, however, not every efficient output can be achieved by simple centralized pricing decisions or by competitive market pricing processes.

3. Moreover, it is possible in the integer programming case that there exists no hyperplane which separates the feasible lattice points from those which are preferred to or indifferent with the optimal lattice point. In other words, there may exist no set of prices which simultaneously makes the optimal point, Q, the most profitable among those that can be produced and the cheapest among those that consumers consider to be at least as good as Q. That is, at any set of prices either producers will try to make, or consumers will demand, some other output combination.

It should be observed, in conclusion, that these limitations on the price system in the integer programming case should not be entirely surprising. For, as has already been indicated, cases of increasing returns to scale can, at least in principle, be reduced to integer programming problems. And in such cases it has long been recognized that the price system runs into difficulties.

5. NONCONVEX FEASIBLE REGIONS AND CONCAVE PROGRAMMING

Several of the nonnumber-theoretical applications of integer programming should be clear to the economist. The choice of magnitudes of indivisible outputs obviously calls for integer programming, though here ordinary programming methods will often do as an approximation (e.g., an answer which calls for a retailer to carry 47.9 automobiles in stock may reasonably be taken to indicate that 48 is the optimal car inventory). Such an easy compromise is not available in "yes or no" problems like the traveling salesman problem or the following problem of "choosing the largest harmonious expedition." Suppose an expedition is to be made up from n candidates with the condition that no two candidates who can't get along with each other are to be taken. Assigning a variable x_i to the ith candidate, we shall interpret a value of 0 to mean that that candidate is included in the expedition, a value of 1 to mean that he is excluded. The variable x_i is to be restricted to these two values. The problem of constructing the largest harmonious expedition then is the problem of minimizing $\sum_{i=1}^{n} x_i$, the number left out subject to restrictions

$$x_i + x_j \geq 1$$

for all pairs i,j of candidates who can't get along. The effect of each such restriction is to insist that at least one candidate in the i,j pair is left out. It is not hard to see that if the problem is solved as an integer programming problem, the variables in the minimum solution will not only be integers, but actually 0's and 1's, for if any larger integer is included in the minimal solution it could be decreased to 1 without violating any constraints. This would produce a solution with a still smaller objective function. Thus the problem can be solved as an integer programming problem, but it will be noted that an ordinary linear programming solution involving fractions, has no obvious meaning.

Less obvious are the more general applications of integer programming to nonconvex feasible regions and to concave programming problems. An example will now be described briefly.

It will be recalled that fixed costs are defined as costs which do not vary with the magnitude of some operation (at least within limits), and that these costs can therefore be escaped only by closing the operation down altogether. We will see now what computational problems expenses of this type can produce.

Figure 6 represents part of the profit function of a multi-branch firm showing how company profits will vary when the scale of operation of one of its branches, B, varies, the outputs of all other branches being given. This relationship is profit curve TRR'.

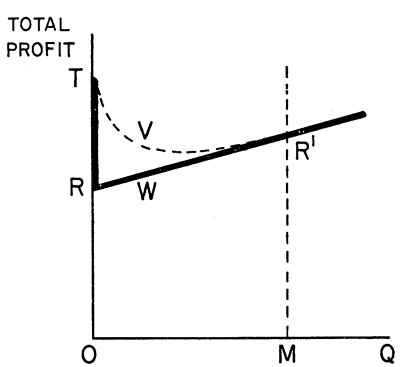

FIGURE 6

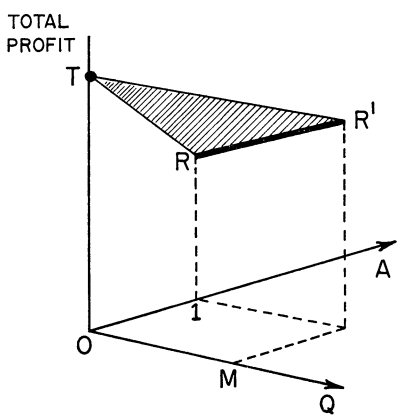

FIGURE 7

As the diagram shows, if this branch is kept in operation, the larger its output, Q, the larger will be the firm's profits (RR' slopes upill toward the right). But in the case shown, if the branch goes out of operation altogether, the fixed costs which it escapes are so large that company profits will suddenly jump from R to T. In fact (assuming that there is some upper limit, OM,

to the demand for its product) even if the branch produces every bit that it can sell, the profit contribution of this branch will not suffice to cover the fixed cost, because point R', whose height represents profit at the maximum saleable output, lies below T, where OT represents company profit when the plant is closed down altogether.

We see, then, that point R' is a local maximum but T is the global maximum. However, any computation which tells us to go uphill along the profit curve will move us in the wrong direction. Even at a point like W which is very close to R there is not the slightest hint in the shape of the curve that profits can be increased by *reducing* output. This is a particularly nasty feature of the fixed charges problem. An ordinary increasing (marginal) returns profit curve (a convex objective function maximization problem), such as curved line TVR, will at least indicate the direction of the global maximum point when we get close enough to it—at point V going uphill takes us toward global optimum T, even if starting further to the right the "go uphill" rule would take us in the wrong direction.

It is, of course, only because we are dealing with a multi-branch firm that our problem is really difficult. As a result, even our graph is likely not to give us the right answer. Perhaps it is best not to close our branch B after all. Instead it might be better to close some other branch, C, and save the fixed charges at C, meanwhile serving C's former customers from B, for this increases the maximum demand for branch B's products and so permits us a higher move along our profit curve to the right of point R'. With a large number of branches the problem of examining the possibilities case by case, to decide how many and which to close, leads us into an enormous problem of permutations and combinations which rapidly grows astronomical. A more systematic computation is required.

A similar problem arises in the search for optimal investment criteria. Suppose, for example, that a country has limited investment funds to be divided between two competing projects. The first yields a low rate of return but has low fixed costs of entry into production, and the reverse is true of the second project. Which of the projects should be chosen will clearly depend on the magnitude of the fixed costs.

The role of integer programming in such a problem is easily represented schematically. For this computation it is necessary to introduce an artificial variable, A. In the three dimensional diagram, Figure 7, point T from the original profit function is placed where $A = 0$, while line RR' is moved to where $A = 1$. The three points T, R, and R' are then connected by the plane TRR' which can now serve as the feasible portion of an artificial *linear* programming objective function. But if we include the constraint $A \geq 0$ and $A \leq 1$ in the problem and require that A take only integer values it is clear that we can only have either $A = 0$ or $A = 1$. We can end up only at

point T or on line segment RR', i.e., we must remain somewhere on the original profit curve TRR' of Figure 6. Thus by use of integer programming we have been able to substitute for our original fixed charges problem another ordinary linear programming problem which gives the same answers.[11]

In principle, this translation can be made for all of the company's branches at once and so the entire problem can be transformed into one large linear integer programming problem and thus be solved. Unfortunately, in practice this has not so far proved practical for even moderately large scale problems where the number of artificial variables which must be added can make the computation prohibitively time consuming and expensive.

6. THE SIMPLEX CALCULATION: A CONDENSED FORM

Before giving numerical examples of the integer programming computation it is convenient to call attention to a number of short cuts in the simplex computation in which we follow the work of A. W. Tucker.

The problem is set up in the form

$$\left\{\begin{array}{l} \max z = a_{o,o} + a_{o,1}(-t_1) + \ldots + a_{o,n}(-t_n) \\ \quad \text{subject to} \\ t'_1 = a_{1,o} + a_{1,1}(-t_1) + \ldots + a_{1,n}(-t_n)\,, \\ \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ t'_m = a_{m,o} + a_{m,1}(-t_1) + \ldots + a_{m,n}(-t_n)\,, \end{array}\right. \tag{6.1}$$

with all of the variables required to take nonnegative values. First we note that if all the elements in the first column (the constant terms) are nonnegative a (basic) feasible solution is given by[12]

$$t'_1 = a_{1,o}, \ldots, t'_m = a_{m,o}, t_1 = \ldots = t_n = 0\,. \tag{6.2}$$

Thus the system is said to be *primal feasible* if for all $i \neq 0$ we have $a_{i,o} \geq 0$.

[11] Note that as described this is a "mixed" problem in which some but not all of the variables are required to be integer values. The MIF method does not apply directly to such problems. The difficulty can be evaded, at least in principle, by measuring outputs in very small units and taking their optimal *integer* values as approximations to their true optimal values. By making the units of measurement small enough this approximation can, in principle, clearly be made as close as possible, though we do not yet have enough computing experience with the MIF algorithm to know how rapidly it converges when dealing with the large numbers which are likely to result. There has also been some promising work on the mixed problem. Cf. Beale [**2**] and Gomory [**8**].

[12] We note again that in this computation the basic variables are expressed as functions of the variables outside the basis. This is the reverse of the more usual viewpoint. It permits us to solve directly for the values of the basic variables at the relevant corner, as shown.

Similarly, for obvious reasons, it is said to be dual feasible if the coefficients of the objective function $a_{o,j}$ are nonnegative.

It should be clear by inspection that if the system is transformed into a form that is both primal and dual feasible then (6.2) is also an optimal solution. The primal simplex method then proceeds by starting with the problem in primal feasible form and then transforming it by a sequence of steps in a way which leaves the $a_{i,o}$ nonnegative while increasing the $a_{o,i}$ until (6.1) becomes both primal and dual feasible.

To describe the simplex steps consider the following two illustrative constraint equations of the problem

$$t_1' = a_{1,o} + a_{1,1}(-t_1) + a_{1,2}(-t_2) + a_{1,3}(-t_3) ,$$
$$t_2' = a_{2,o} + a_{2,1}(-t_1) + a_{2,2}(-t_2) + a_{2,3}(-t_3) .$$

Suppose that the computations have reached a stage where there is to be a change in basis from c_1, c_2 to, say, c_1, t_3 (we say that we *pivot* on coefficient $a_{2,3}$, i.e., we replace c_2 by t_3 in the basis). Then we solve for t_2 by dividing the second equation through by $a_{2,3}$ and substitute the result into the first equation to obtain

$$t_1' = \left(a_{1,o} - \frac{a_{1,3}}{a_{2,3}} a_{2,o}\right) + \left(a_{1,1} - \frac{a_{1,3}}{a_{2,3}} a_{2,1}\right)(-t_1) + \left(a_{1,2} - \frac{a_{1,3}}{a_{2,3}} a_{2,2}\right)(-t_2) - \frac{a_{1,3}}{a_{2,3}}(-t_2') .$$

More generally, the reader may readily verify that a pivot on element $a_{i,j}$ will replace element $a_{v,w}$ ($v \neq i$, $w \neq j$) by

$$a_{v,w} - \frac{a_{i,w}}{a_{i,j}} a_{v,j} .$$

A slight extension of the argument leading to the preceding equation shows that any pivot step will lead to the following changes in the values of the coefficients in (6.1) (the elements of the matrix of the system):

(6.3)
a) The pivot element, $a_{i,j}$, will be changed to $a'_{i,j} = 1/a_{i,j}$.
b) Any other element $a_{v,j}$ in the pivot column j will be changed to $a'_{v,j} = -a_{v,j}/a_{i,j}$.
c) Any element $a_{i,w}$ in the pivot row i will be changed to $a'_{i,w} = a_{i,w}/a_{i,j}$.
d) Any other element $a_{v,w}$ will, by the preceding argument, be changed to $a_{v,w} - a_{i,w}a_{v,j}/a_{i,j}$.

One feature of the method we are using is that the same transformation (6.3) is applied to the objective function as to the constraints. This has the effect of expressing the objective function always in terms of the non-basic variables (the t_i) only, so that we always end up with an objective function

of the form taken in (6.1). As a result, as soon as the system is transformed into both primal and dual feasible form the solution (6.2) is immediate.

The primal simplex method seeks to increase the value of the objective function, $a_{o,o}$, and so, like the dual simplex method, it pivots in a column with the first element negative, but unlike the dual method, it always pivots on a positive element. For with $a_{i,j} > 0$, $a_{o,j} < 0$ and $a_{i,o} \geq 0$ (by primal feasibility) it follows from (6.3d) that $a'_{,o} \geq a_{o,o}$ as required. The corresponding result for the dual simplex method in which we seek to reduce (minimize) the value of the objective function, is obvious.

The procedure used in the solution of the integer programming problem is, then, the following:

A. *The condensed form primal simplex calculation.*

1. Set up the matrix (simplex tableau) for a primal feasible system (6.1).
2. Choose a column, j, with the first element $a_{o,j}$ negative (in the illustrative computation we always choose the largest such element in absolute value).
3. Choose as the pivot that positive element in this column which minimizes $a_{i,o}/a_{i,j}$. (This element is chosen to keep the next matrix dual feasible. For suppose instead we chose as pivot $a_{k,j}$ such that $a_{k,o}/a_{k,j} > a_{i,o}/a_{i,j}$. Then by (6.3d) we would have $a'_{i,o} = a_{i,o} - a_{i,j}a_{k,o}/a_{k,j} < 0$.)
4. Transform the matrix in accord with (6.3).
5. Repeat steps 1–4 until the matrix becomes dual feasible so that optimal solution (6.2) applies.

B. *The addition of an MIF constraint.*

If solution (6.2) contains noninteger values, form an additional constraint (2.5). This is done by choosing a row j and writing an additional constraint with the same variables but whose coefficients are the *negative* fractional parts of the corresponding elements in row j. (In this computation we choose row j to be the one whose first element has the largest fractional part for reasons indicated at the end of Section 2, above.)

C. *The dual simplex calculation.*

The problem is now in dual feasible form (since the last step of the primal simplex computation put it in that form). However, it is no longer in primal feasible form since the new constraint (2.5) enters the negative element $-f_{i,o}$ into the first column. The new optimum is therefore found most conveniently by the dual simplex method which differs from the primal method only in the choice of pivot element. Here select we a *row* i whose first element $a_{i,o}$ is negative and from that row select we a *negative* pivot element $a_{i,j}$ which minimizes $-a_{i,o}/a_{i,j}$. Once the new optimal solution is

found, if it contains noninteger elements we repeat steps B and C until an optimal integer solution is found.

7. EXAMPLE

$$\begin{aligned} \text{Maximize} \quad & z = 4x_1 + 5x_2 + x_3 \\ \text{subject to:} \quad & 3x_1 + 2x_2 \leq 10, \\ & x_1 + 4x_2 \leq 11, \\ & 3x_1 + 3x_2 + x_3 \leq 13. \end{aligned}$$

Introducing slack variables $\bar{x}_1$, $\bar{x}_2$, $\bar{x}_3$ we obtain the following sequence of simplex tableaux, where the asterisk indicates the pivot element and the arrow indicates the row from which the new inequality is formed:

	1	$-x_1$	$-x_2$	$-x_3$
$z =$	0	−4	−5	−1
$\bar{x}_1 =$	10	3	2	0
$\bar{x}_2 =$	11	1	4*	0
$\bar{x}_3 =$	13	3	3	1

TABLEAU 1

	1	$-x_1$	$-\bar{x}_2$	$-x_3$
$z =$	$13\frac{3}{4}$	$-2\frac{3}{4}$	$1\frac{1}{4}$	−1
$\bar{x}_1 =$	$4\frac{2}{4}$	$2\frac{2}{4}$*	$-\frac{2}{4}$	0
$x_2 =$	$2\frac{3}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0
$\bar{x}_3 =$	$4\frac{3}{4}$	$2\frac{1}{4}$	$-\frac{3}{4}$	1

TABLEAU 2

	1	$-\bar{x}_1$	$-\bar{x}_2$	$-x_3$
$z =$	$18\frac{7}{10}$	$1\frac{1}{10}$	$-\frac{7}{10}$	−1
$x_1 =$	$1\frac{8}{10}$	$\frac{4}{10}$	$-\frac{2}{10}$	0
$x_2 =$	$2\frac{3}{10}$	$-\frac{1}{10}$	$\frac{3}{10}$	0
$\bar{x}_3 =$	$\frac{7}{10}$	$-\frac{9}{10}$	$-\frac{3}{10}$	1*

TABLEAU 3

	1	$-\bar{x}_1$	$-\bar{x}_2$	$-\bar{x}_3$
$z =$	$19\frac{4}{10}$	$\frac{2}{10}$	$\frac{4}{10}$	1
$x_1 =$	$1\frac{8}{10}$	$\frac{4}{10}$	$-\frac{2}{10}$	0
$x_2 =$	$2\frac{3}{10}$	$-\frac{1}{10}$	$\frac{3}{10}$	0
$\rightarrow x_3 =$	$\frac{7}{10}$	$-\frac{9}{10}$	$-\frac{3}{10}$	1
$s_1 =$	$-\frac{7}{10}$	$-\frac{1}{10}$	$-\frac{7}{10}$*	0

TABLEAU 4 (L.P. optimal)[13]

	1	$-\bar{x}_1$	$-s_1$	$-\bar{x}_3$
$\rightarrow z =$	19	$\frac{1}{7}$	$\frac{4}{7}$	1
$x_1 =$	2	$\frac{3}{7}$	$-\frac{2}{7}$	0
$x_2 =$	2	$-\frac{1}{7}$	$\frac{3}{7}$	0
$x_3 =$	1	$-\frac{6}{7}$	$-\frac{3}{7}$	1
$\bar{x}_2 =$	1	$\frac{1}{7}$	$-1\frac{3}{7}$	0
$s_2 =$	0	$-\frac{1}{7}$*	$-\frac{4}{7}$	0

TABLEAU 5 (integer solution)

[13] In the original coordinates the inequality $s_1 \geq 0$ becomes the new integer inequality (see Appendix A) $-\frac{7}{10} + \frac{1}{10}(10 - 3x_1 - 2x_2) + \frac{7}{10}(11 - x_1 - 4x_2) \geq 0$, i.e., $x_1 + 3x_2 \leq 8$.

	1	$-s_2$	$-s_1$	$-\bar{x}_3$
z =	19	1	0	1
x_1 =	2	3	2	0
x_2 =	2	−1	−1	0
x_3 =	1	−6	3	1
$\bar{x}_2$ =	1	1	−2	0
$\bar{x}_1$ =	0	−7	4	0

TABLEAU 6 (all integer matrix)

APPENDIX A

SOME PROPERTIES OF THE ADDED INEQUALITIES

To fill the gap in the proof in Section 3 that the average price of the original capacities is reduced by the artificial constraints (2.5) we must show that the $Q_i \geq 0$. We can assume this for the Q_i of the original problem because of their capacity interpretation, but it remains to be shown for Q_i, $i > m$. We adopt a method of proof that brings out some interesting properties of the added inequalities.

If we express the new variables, t, in (1.4), above, in terms of the original variables x in (1.1) and substitute the result into the new inequalities, these inequalities are rewritten entirely in terms of the original variables. We will show now that they are then all-integer inequalities, i.e., all coefficients and constants are integers. To see this we will first assume that the inequality under consideration is the first new inequality to be added. It is derived from an equation

$$t_i' = a_{i,o} + \sum_{j=1}^{n} a_{i,j}(-t_j)$$

where the variables, t, are either x's or slack variables of the original problem. For each t_j on the right we substitute its original expression in terms of the x_j, i.e., if t_j is $x_{k(j)}'$ the kth slack, we substitute $Q_k - \sum_{i=1}^{n} a_{k,j}^{*} x_j$ $(= x_{k(j)}')$ and if t_j is some $x_{i(j)}$ we simply substitute $x_{i(j)}$. We thereby obtain an expression giving t_i' in terms of the x_j. As the expression for any variable in terms of the original non-basic variables is unique, this be one of the original all-integer equations if t_i' is a slack, or the expression $x_i = x_i$ if t_i' is one of the x's. In any event, the new right hand side is all integer. If the same process is applied to the new equation

$$(2.5) \qquad s_i = -f_{i,o} - \sum_{j=1}^{n} f_{i,j}(-t_j) = \left(u_{i,o} + \sum_{j=1}^{n} u_{i,j}(-t_j)\right) - \left(a_{i,o} + \sum_{j=1}^{n} a_{i,j}(-t_j)\right)$$

where the $u_{i,j}$ are the integer parts of the $a_{i,j}$, the result is again an all-integer right hand side. This is obviously so for the all-integer expressions involving the $u_{i,j}$, and we have just shown the same thing for the second parenthesis. We conclude that *the new inequality $s_i \geq 0$, is an all-integer inequality when it is expressed in the original variables.*

Looking at the substitution process in more detail, using J for the set of indices j whose t_j are original variables $x_{i,j}$, we have

$$s_i = -f_{i,o} + \sum_{j \in J} f_{i,j} x_{k(j)} + \sum_{j \in J'} f_{i,j} \left(Q_{i,j} - \sum_{k=1}^{n} a^*_{i(j),k} x_k \right) \tag{A.1}$$

or, assembling constant terms and the coefficients of the various x_k, the inequality $s_i \geq 0$ becomes

$$-f_{i,o} + \sum_{j \in J'} f_{i,j} Q_{i,j} \geq - \sum_{j \in J} f_{i,j} x_{k(j)} + \sum_{k=1}^{n} \left(\sum_{j \in J'} f_{i,j} a^*_{i(j)k} \right) x_k .$$

The left hand side contains only nonnegative terms except for $-f_{i,o}$, hence it is > -1. We have already shown, however, that all terms are integers. Hence the left hand side must be ≥ 0, and this left side is the Q_i of the new inequality.

In this argument we assumed we were dealing with the first added inequality. However, now that we have established this inequality as an all integer one with nonnegative Q_i we can go on without any difficulty to the second, third, etc.

We obtain one more piece of information by a similar argument. Suppose that the coefficients $a_{i,k}$, $i = 1, \ldots, m$, of some variable x_k are all nonnegative in the original problem, *then the coefficients of x_k are also nonnegative in the new inequalities*. For the coefficient is either

$$\sum_{j \in J} f_{i,j} a^*_{i(j),k}$$

or, if x_k is one of the $x_{k,j}$, $j \varepsilon J$, the same expression with the additional term $-f_{i,j'}$. If the $a^*_{i,j,k}$ are all ≥ 0, as we assume, this term, too, is > -1 and an integer and hence ≥ 0.

In particular, if the original inequalities involved only nonnegative terms, this is also true of the added inequalities if these are written in terms of the original variables.

APPENDIX B

IMPUTING BACK THE PRICES OF THE ARTIFICIAL CONSTRAINTS

It will be observed that each parenthesis on the right in equation (A.1) of Appendix A, if set ≥ 0, gives one of the original inequalities (1.1). Thus, *each of the new inequalities, $s_i \geq 0$, differs by a constant, $-f_{i,o}$, from a weighted sum of the original constraints.* (Note that this statement refers both to the explicitly given inequalities such as $Q_i - \sum_{j=1}^{n} a_{i,j} x_j \geq 0$ and to the implicit final output nonnegativity inequalities, $x_j \geq 0$.)

This suggests that the prices associated with the new inequalities can be distributed back to the original inequalities which compose them. This is not hard to do. To simplify the exposition we will first consider the case where only one new inequality has been added. This extends easily to the general case.

Suppose, then, that on solving the integer programming problem we obtain prices π_i for all the original goods (capacities) and for the artificial capacities. We already have "prices" for the final goods—these are the unit profits of the activities—the coefficients of the activity levels in the objective function.

To display this price information symmetrically we write the original inequalities augmented by the conditions, $x_j \geq 0$, together with the associated prices π_i:

$$\begin{aligned}
\sum_{j=1}^{n} a^*_{1j}\, x_j &\leq Q_1 \quad \pi_1 \\
\sum_{j=1}^{n} a^*_{2,j}\, x_j &\leq Q_2 \quad \pi_2 \\
&\vdots \qquad\qquad \vdots \\
\sum_{j=1}^{n} a^*_{m,j}\, x_j &\leq Q_m \quad \pi_m \\
-x_1 &\leq 0 \quad \pi_{m+1} \\
-x_2 &\leq 0 \quad \pi_{m+2} \\
&\;\;\cdot \\
&\;\;\;\cdot \\
-x_n &\leq 0 \quad \pi_{m+n}\,.
\end{aligned}$$

The additional inequalities, as (A.1) shows, can be thought of as being obtained by adding together nonnegative multiples of the preceding inequalities and then reducing the right hand side by a certain constant to obtain the new integer inequality

$$\sum_{j=1}^{n} a^*_{m+n+k}\, x_j \leq Q_{m+n+k}\, \pi_{m+n+k}\,.$$

For simplicity let us suppose further that the integer solution has been obtained after the addition of our single new inequality. Extension of the method to the more usual situation will not require any additional effort.

We have then prices π_i, $i = 1, \ldots, m + n + 1$, and $m + n + 1$ inequalities

$$\sum_{j=1}^{n} a^*_{i,j}\, x_j \leq Q_j \qquad (i = 1, \ldots, m+n+1)$$

where we include the inequalities $-x_j \leq 0$.

The prices obtained from the solution have the usual linear programming property $\sum_{i=1}^{m+n+1} \pi_i\, a^*_{i,j}\, x_j \leq \pi_j x_j$,
i.e.,

$$\sum_{i=1}^{m+n+1} \pi_i\, a^*_{i,j} \leq 0 \qquad (j = 1, \ldots, n) \tag{B.1}$$

with equality required for all j having $x_j \neq 0$, this last requirement of equality being equivalent to

$$\sum_{i=1}^{m+n+1} \pi_i\, Q^*_i = 0 \tag{B.2}$$

with Q^*_i the amount of the ith capacity or input good used up or, if i is a final good, Q^*_i represents the amount produced.

(B.1) represents the requirement that cost exceed or at best equal the value of the final good produced, and (B.2) asserts that at the prices π_i the value of input goods used equals the value of output goods produced. This is equivalent to the requirement

that equality must hold in (B.1) in all cases where the final good, j, is actually produced in a positive amount, for

$$0 = \sum_{i=1}^{m+n+1} \pi_i Q_i^* = \sum_{i=1}^{m+n+1} \pi_i \sum_{j=1}^{n} a_{i,j}^* x_j ,$$

$$0 = \sum_{j=1}^{n} \left(\sum_{i=1}^{m+n+1} \pi_i a_{i,j}^* \right) x_j .$$

Since each parenthesis is ≤ 0, the only way for the zero total to be achieved is for $(\sum_{i=1}^{m+n+1} \pi_i a_{i,j}^*)$ to be zero for each nonzero x_j.

In this situation in the case of an input good it makes no difference in (B.2) if we use the Q_i^*, the amount used, or the Q_i, the amounts available, as these quantities will differ only for goods of price zero.

Let us now describe a procedure for imputing the prices of the artificial constraints back to the original constraints of which they are linear combinations. If we denote the row vector of coefficients $(a_{i,1}^*, a_{i,2}^*, \ldots, a_{i,n}^*)$ by R_i, we know by (A.1) that R_{m+n+1} is a nonnegative combination of the preceding R_i, i.e.

$$R_{m+n+1} = \sum_{i=1}^{m+n} g_i R_i .$$

(Here, because only one inequality has been added, the nonzero g_i are the f_j used in forming the new inequality.) (B.1) requires

$$0 \geq \sum_{i=1}^{m+n+1} \pi_i R_i = \pi_{m+n+1} R_{m+n+1} + \sum_{i=1}^{m+n} \pi_i R_i = \sum_{i=1}^{m+n} (\pi_{m+n+1} g_i + \pi_i) R_i .$$

This equation shows that if new (increased) prices $\pi_i' = \pi_{m+n+1} g_i + \pi_i$ are assigned to the original goods, and the additional inequality disregarded, i.e., given zero price, the condition of profitless production is still maintained. Also since $\sum_{i=1}^{m+n+1} \pi_i R_i = \sum_{i=1}^{m+n} \pi_i' R_i$, i.e., $\sum_{i=1}^{m+n+1} \pi_i a_{i,j}^* = \sum_{i=1}^{m+n} \pi_i' a_{i,j}^*$ = profit per unit of final good j for every j, the same final goods as before are made at zero loss, so the property (B.2) still holds with recomputed prices i.e., $\sum_{i=1}^{m+n} \pi_i' Q_i^* = 0$.

To obtain these prices in the case where more than one inequality has been added one takes the last added inequality, which represents a known weighted combination of earlier inequalities, and uses the explicit expression for this inequality to generate new prices just as above. Having thus gotten rid of this inequality one proceeds to the next to last, and so on until only original inequalities remain. The prices obtained by this process will have another desirable property:

Result (1). An original input good receiving a zero price will always be a free good in the sense that if unlimited amounts of it were available, the output of final goods would still not be affected.

To see this we consider the computation which has been gone through to obtain the original dual prices noting first that a nonartificial input with a zero recomputed price must always also have a zero dual price because the price recomputation process never lowers the price of such an item.

The original computation can be repeated step by step with the zero-priced inequality

removed simply by regarding the slack x_i' of that inequality as an unrestricted variable rather than a nonnegative one.

When the computation has been completed the zero-priced slack variable x_i', even if it is non-basic, must, as we have noted, have a zero dual price (zero coefficient in the top row). In either of these cases, despite the presence of an unrestricted variable, the final tableau still gives the old optimal solution to the problem because of the nonnegativity of the $a_{i,o}$ and $a_{o,j}$, provided only that we still know that those non-basic s_i having nonzero coefficients in the top row, i.e., nonzero prices before any redistribution of prices, are still required to be nonnegative. Now the nonnegativity of each s_i stems from its being given be an equation

$$s_i = -f_i - \Sigma f_j(-t_j) .$$

Since s_i is an integer, and *if the t_j are nonnegative*, we have $s_i \geq -f_i$, and hence $s_i \geq 0$. If, however, x_i' was among the t_j accompanied by a nonzero f_j, this reasoning fails, and we can no longer require $s_i \geq 0$ and the final tableau no longer gives an optimal solution. But if the ith inequality received a final redistributed price of 0, this undesirable situation can not have occurred. For if s_i received a nonzero price π, then, upon redistributing, the inequality with slack x_i' would have received an increase in price of πf_i and so its price could not be zero after redistribution.

The situation is somewhat more complicated if s_i is originally expressed in terms of other earlier s-variables whose nonnegativity has been endangered by the unrestricted sign of x_i'. However, the same argument, though it requires more words, does go through step by step.

Result (2). A second conclusion is the following. If there is some set of n original inequalities such that these n alone yield the same integer solution as does the full set of inequalities, then it is possible to redistribute prices in such a way that the prices of all input goods are simply the ordinary linear programming prices, i.e., the prices obtained for the goods if the program involving only these n inequalities were solved as an ordinary noninteger linear programming problem, omitted goods receiving zero price.

A consequence of result (2) is the fact that, in general, the converse of result (1) is not valid. It is not always true that if an inequality can be removed without changing the solution, i.e., if it represents a free good, that this good will receive a zero price. The connection is illustrated in Figure 8 in which L_1, L_2, L_3 represent constraints and the dashed line is an isoquant of the objective function. It is clear that restrictions L_1 and L_2 alone determine the solution P, and consequently, according to result (2), it is possible to redistribute prices so that L_1 and L_2 receive the ordinary (usually positive) linear programming prices that would result from an ordinary noninteger programming problem with L_3 omitted, and L_3 receives a zero price. However, in the problem L_2 is a free good in the sense that, since L_1 and L_3 also yield the same solution P, unlimited availability of the good involved in the restraint L_2 would not alter the solution.

This difficulty also arises in ordinary linear programming whenever several subsets of inequalities separately determine the answer. However, in ordinary linear programming this is comparatively rare since it must involve degeneracy (this is precisely what is meant by degeneracy). Such a situation is shown in Figure 9.

We now take up the proof of result (2). To obtain the prices in question we simply ignore the constraints other than the N singled out, and proceed to solve the problem first as an ordinary noninteger linear programming problem, and then as usual, by

adding new inequalities to obtain the integer solution. At the solution point of the ordinary linear programming problem, all N inequalities are satisfied as equalities, i.e. all the slack variables are nonbasic (we may regard x as being the slack of the inequality $-x \leq 0$, for if we introduce an x', with $-x + x' = 0$, we have $x = x'$). This

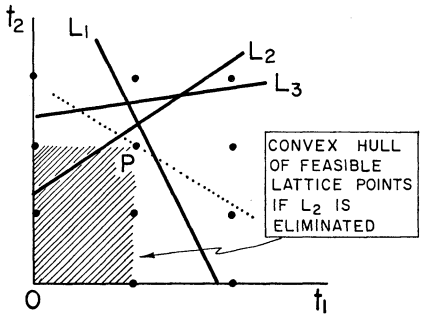

FIGURE 8

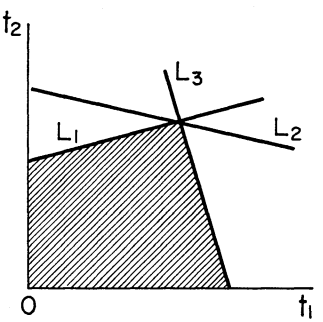

FIGURE 9

point may well not satisfy the inequalities that we are temporarily disregarding. This fact makes no difference, however, when it comes to generating the new inequalities whose validity is not affected. Of course when we reach the final integer solution, which is the same for the problem with or without these inequalities, the disregarded inequalities will be satisfied by hypothesis.

Now on redistributing the prices we distribute them back to exactly n inequalities, the original n inequalities. In other words we express each new variable in terms of the n slacks x'_j, $j = 1, \ldots, N$:

$$s_i = -g_{i,o} + \sum_{j=1}^{n} g_{i,j} x'_j . \tag{B.3}$$

Thus all the final non-basic variables are expressed this way.

If each non-basic variable (or the inequality it represents) receives a price π_i in the usual linear programming way, then, on distributing the prices back to the x'_j, they receive prices

$$\pi'_j = \sum_{i=1}^{N} \pi_i g_{i,j} . \tag{B.4}$$

To see that these are in fact the ordinary linear programming prices we note that the final set of equations can be augmented to express all the variables in terms of the non-basic ones. The basic variables are already given in terms of the non-basic ones, and the non-basic ones can certainly be given in terms of themselves. Having then an expression for all variables in terms of the non-basic ones, we substitute (B.3) for the s_i to obtain an expression for all variables in terms of the x'_i.

The z-equation of the final tableau

$$z = a^*_{o,o} + \sum_{i=1}^{n} \pi_i (-s_i)$$

becomes

$$z = \left(a^*_{o,o} + \sum_{i=1}^{n} \pi_i g_{i,o}\right) + \sum_{j=1}^{n} \left(\sum_{i=1}^{n} \pi_i g_{i,j}\right) (-x'_j) .$$

Note that the coefficient of $(-x_j')$ is the price given by (B.4). Now the expression for the variables in terms of the non-basic set x_j' is unique, so all the coefficients must be identical with those obtained in solving the n inequalities of an ordinary linear programming problem so that the ordinary linear programming price and the redistributed price coincide for input goods.

REFERENCES

[1] ARROW, KENNETH J.: "An Extension of the Basic Theorems of Classical Welfare Economics," *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, 1951.

[2] BEALE, E. M. L.: "A Method of Solving Linear Programming Problems When Some but Not All of the Variables Must Take Integral Values." Statistical Techniques Research Group Technical Report No. 19, Princeton, N. J., July, 1958.

[3] DANTZIG, GEORGE B.: "On the Significance of Solving Linear Programming Problems with Some Integer Variables," (Ditto) The RAND Corporation, Paper P-1486, September, 1958.

[4] DEBREU, GERARD: "The Coefficient of Resource Utilization," *Econometrica*, Vol. 19, July, 1951.

[5] DORFMAN, ROBERT, PAUL A. SAMUELSON, AND ROBERT M. SOLOW: *Linear Programming and Mathematical Analysis*, McGraw-Hill, New York, 1958.

[6] GOMORY, RALPH E.: "Outline of an Algorithm for Integer Solutions to Linear Programs," *Bulletin of the American Mathematical Society*, Vol. 64, September, 1958.

[7] ———: "An Algorithm for Integer Solutions to Linear Programs," (mineographed) Princeton—I.B.M. Mathematics Research Project, Technical Report No. 1, November, 1958.

[8] ———: "A Method for the Mixed Integer Problem," in preparation as a RAND Corporation report.

[9] KOOPMANS, TJALLING C., ed.: *Activity Analysis of Production and Allocation*, Cowles Commission Monograph 13, Wiley, New York, 1951.

[10] ———: *Three Essays on the State of Economic Science*, McGraw-Hill, New York, 1957.

[11] KOOPMANS, TJALLING C. AND MARTIN BECKMANN: "Assignment Problems and the Location of Economic Activities," *Econometrica*, Vol. 25, January, 1957.

[12] LAND, A. H. AND A. G. DOIG: "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, this issue.

[13] MARKOWITZ, HARRY M. AND ALAN S. MANNE: "On the Solution of Discrete Programming Problems," *Econometrica*, Vol. 25, January, 1957.