MATHEMATICAL PROGRAMMING

R. E. GOMORY, Thomas J. Watson Research Center, IBM

Mathematical programming is the name used for the study of a variety of maximization problems. In these problems, maxima can occur on the boundaries as well as in the interior of a region. In many cases the maximization is actually over a finite set of points. When this happens it is clear that: 1) the usual derivative criteria for a maximum, as used in the calculus, do not apply; and 2) that the emphasis must be on algorithms for actually obtaining maxima rather than on existence, for existence is trivial in these finite cases. Both these observations do, in fact, apply to most of mathematical programming and the second is the reason why the subject is intimately bound up with the use of computers.

In what follows we will discuss two of the most active areas of mathematical programming: first, linear programming, and secondly, and very sketchily, dynamic programming.

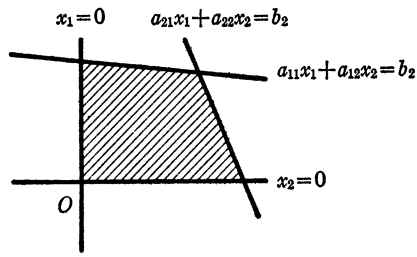

FIG. 1

Linear programming. The maximization problem here is to maximize a linear form subject to linear inequality constraints. The basic problem can be written

$$\max \sum_{j=1}^{j=N} c_j x_j, \qquad \sum_{j=1}^{j=N} a_{ij}x_j \leqq b_i, \qquad i = 1, \cdots, M, \tag{1}$$

$$x_j \geqq 0, \qquad j=1, \cdots, N,$$

or vectorially,

$$\max \; c \cdot x, \quad a_i \cdot x \leqq b_i, \qquad i = 1, \cdots, M+N.$$

Here the last N a_i are negative unit row vectors and the last N b_i are 0. A geometrical way of looking at the problem is illustrated in Figure 1. Each inequality constraint confines the variables to a closed half space, the intersection of these half spaces gives a polyhedron of values satisfying the inequalities. This polyhedron is called the feasible region. The maximization problem is to find that point of the feasible region which maximizes the "objective function" $\sum c_j x_j$.

Maximization criterion. The usual condition for the free maximum of a function at an interior point is that the gradient of the function should vanish. If there are also equality conditions to be satisfied, the condition for a maximum is, roughly, that the gradient should point in a direction that is normal to the equality surfaces. More precisely, the gradient should be a linear combination of the normals to these constraint surfaces—this is the usual Lagrangian condition. When dealing with linear inequalities the condition is similar. The function will be maximal at $\bar{x}$ if the gradient at $\bar{x}$ points into a region forbidden by the inequalities. This means that it should be a nonnegative combination of the normals of those (and only those) inequality constraints that are satisfied as equalities at $\bar{x}$. See Figure 2. Thus at $\bar{x}$, the maximum point of (1), there will be an $M+N$ vector $\bar{y}=(\bar{y}_1, \cdots, \bar{y}_{M+N})$ satisfying

$$\text{(2a)} \qquad c = \sum_{i=1}^{i=M+N} a_i \bar{y}_i, \ \bar{y}_i \geqq 0 \qquad \text{and} \qquad \text{(2b)} \qquad \bar{y}_i > 0 \Rightarrow \bar{a}_i \cdot \bar{x}_i = b_i$$

and, if at a feasible point $\bar{x}$ there is such a $\bar{y}$, then $\bar{x}$ is maximal.

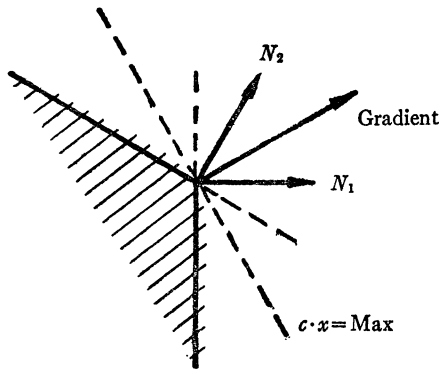

FIG. 2

Duality. Further examination of the maximization condition leads to the notion of duality due to Gale, Kuhn and Tucker [1].

Using (2a) and forming the scalar product with $\bar{x}$ yields, using (2b)

$$\text{(3)} \qquad c \cdot \bar{x} = \sum_{i=1}^{i=M+N} \bar{y}_i a_i \cdot \bar{x} = \sum_{i=1}^{i=M+N} \bar{y}_i b_i.$$

For any other $M+N$ vector y satisfying (2a)

$$c \cdot \bar{x} = \sum_{i=1}^{M+N} y_i a_i \cdot \bar{x} \leqq \sum_{i=1}^{M+N} y_i b_i.$$

Therefore $\bar{y}$ solves the minimization problem

$$\min b \cdot y,\ c = \sum_{i=1}^{M+N} a_i y_i, \qquad y_i \geqq 0. \tag{4}$$

Since $a_{M+1}, \cdots, a_{M+N}$ are negative unit N-vectors, (4) is equivalent to the problem

$$\min b \cdot y, \quad \sum_{i=1}^{i=M} a_i y_i \geqq c, \qquad y_i \geqq 0, \tag{5}$$

in which y is now an M-vector.

(5) is a problem very much like (1). The rows of (1) are the columns of (4), max in (1) has been replaced by min in (5), and the inequality sign reversed in the first set of inequalities. Problem (5) is called the dual of (1). As (3) shows, the max value of (4) and the min of (5), its dual, are numerically equal.

Clearly the dual of the dual is the original problem.

Once we have a solution $\bar{x}$ to (1), we need only represent the gradient at $\bar{x}$ as a nonnegative combination of the normals of the faces on which it lies to obtain a $\bar{y}$ minimizing the dual problem. In most methods of solving (1), this nonnegative representation is usually obtained in the course of recognizing the optimality of $\bar{x}$, so the solution to the dual is usually at hand once the original problem has been solved.

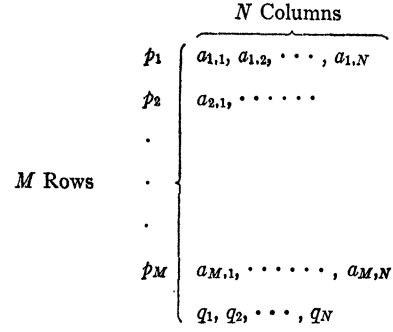

FIG. 3

The theory of zero-sum two-person games provides an interesting illustration of duality. The essential data for a zero sum two-person game is represented in Figure 3. The play of the game is as follows. One player, called the row player, chooses a row, say row i. The other player, called the column player, chooses a column, say column j, without knowing the choice of the row player. The row player then pays the column player an amount a_{ij}. Of course, the row player tries to minimize this amount, the column player to maximize it.

An allowable strategy for the row player is a probability vector $P = (p_1, \cdots, p_M)$. At each play he then chooses the ith row randomly and

with probability p_i. Similarly, the column player has a strategy $Q=(q_1, \cdots, q_N)$. The *expected* payment v, with strategies P and Q is

$$v = \sum_{i=1}^{i=M} \sum_{j=1}^{j=N} a_{ij} p_i q_j. \tag{6}$$

Let us suppose that the row player's object is to choose P so that $v \leqq \theta$ *no matter what his opponent's strategy* Q *is*, and, subject to this condition, to make θ as small as possible.

To find the strategy P that does this we reason as follows: if the column player plays one column only, say the jth, then the expected payment is $\sum_{i=1}^{i=M} p_i a_{ij}$. The best strategy against an opponent confined to single column strategies is the P that minimizes θ subject to

$$\sum_{i=1}^{i=M} p_i a_{ij} \leqq \theta \quad j = 1, \cdots, N \tag{7}$$

and

$$\sum_{i=1}^{i=M} p_i = 1, \; p_i \geqq 0, i = 1, \cdots, M. \tag{8}$$

One can easily verify that (7) implies $v \geqq \theta$ even if a general Q is used by the opponent since the payment then is a weighted sum of numbers each $\leqq \theta$. So (7) and (8) are actually the row player's entire maximization problem.

For simplicity of exposition we will assume here that the smallest possible θ, $\theta_{\min} > 0$, so (7) and (8) can be solved only for $\theta \geqq \theta_{\min} > 0$. Introducing $p_i' = (1/\theta) p_i$, (7) becomes

$$\sum_{i=1}^{i=M} p_i' a_{ij} \leqq 1, \tag{7a}$$

and (8) becomes

$$\sum_{i=1}^{i=M} p_i' = \frac{1}{\theta}, \tag{8a}$$

where $p_i' \geqq 0$, $i=1, \cdots, M$, and $j=1, \cdots, N$. Thus the minimization problem becomes maximize $\sum_{i=1}^{i=M} p_i'$ subject to (7a) which is a problem of the form of (1).

Its dual is min $\sum_{j=1}^{j=N} q_j'$, subject to

$$\sum_{j=1}^{j=N} q_j' a_{ij} \geqq 1, \qquad i = 1, \cdots, M, \tag{9}$$

$q_j' \geqq 0$, $j=1, \cdots, N$. When $1/\beta = \sum_{j=1}^{j=N} q_j'$ and $q_j = \beta q_j'$ are introduced, (9) becomes

$$\begin{aligned} &\max \beta \\ &\sum_{j=1}^{j=N} q_j a_{ij} \geqq \beta \qquad i = 1, \cdots, M, \\ &q_j \geqq 0 \qquad j = 1, \cdots, N, \\ &\sum_{j=1}^{j=N} q_j = 1. \end{aligned} \tag{10}$$

Now (10) is the column player's problem, i.e., choose a Q such that the expected payment is at least β, no matter what the row player's strategy is. Thus, with

the aid of a change of variables, we see that the two players' problems are dual to each other.

Since the max of (8a) equals the min in (9), $1/\theta_{\min} = 1/\beta_{\max}$, so $\theta_{\min} = \beta_{\max} = \bar{v}$.

No matter what the column player does, the row player's expected payment need never exceed $\bar{v}$, if he plays optimally. Similarly, no matter what the row player does, the column player's expectation, if he plays optimally, is always $\geqq \bar{v}$. If both play optimally, the expected payment is exactly $\bar{v}$ which is called the value of the game. Also, because of duality, the strategy Q is usually immediately available, once P is known.

Maximization method. It is geometrically clear, and easily proved, that the maximum in equation (1) will be obtained at one of the vertices of the feasible polyhedron. Thus we are dealing with a finite maximization problem. It is sufficient to search through the vertices of the polyhedron to find the point taking on the maximum value. The simplex method of G. B. Dantzig [2] the usual maximization method of linear programming, inspects only vertices. Instead of looking at all of them, however, it starts at one, moves on to a neighboring vertex that gives a larger value to the function being maximized, and so on until the maximum is reached.

To simplify the discussion of the simplex method, we introduce new variables called "slacks." Slacks represent the difference between the left-hand side of the inequalities of (1) and the right-hand side. When slacks are introduced these equations become

$$\max \sum_{j=1}^{j=N} c_j x_j$$

$$\sum_{j=1}^{j=N} a_{ij} x_j + s_i = b_i \qquad i = 1, \cdots, M, \tag{11}$$

$x_j \geqq 0$, $j = 1, \cdots, N$, $s_i \geqq 0$, $i = 1, \cdots, M$. If s_i is nonnegative, the x's satisfy the ith inequality in (1); consequently the inequalities in (1) can be replaced by equation (11) and the condition that *all* variables appearing are nonnegative. Let us then denote the matrix of all coefficients appearing in the equations (11) by A, and the $M+N$ vector of all variables by y. (11) becomes

$$\max c \cdot y,\ Ay = b,\ y \geqq 0.$$

A vertex is a point where N of the inequalities of (1) are satisfied as equalities. This means that N of the components of y are zero. To obtain the values of the remaining M variables (called basic variables), we simply strike out the columns corresponding to the zero value variables (the nonbasic ones) and solve the remaining M equations in M unknowns. That is, if $\bar{y}$ is the M-vector of basic variables, and B the square matrix formed from the corresponding columns, then $\bar{y} = B^{-1}b$. Thus all components of y are determined, and so in particular x, the

location of the vertex is known. If all $y_i \geqq 0$, the point x satisfies (1) and so is a vertex of the feasible polyhedron; otherwise, it is merely an intersection of N hyperplanes outside the feasible polyhedron.

In carrying out the simplex method we start with a feasible vertex. The next step is to examine neighboring vertices and find those that are: 1) on the polyhedron; 2) yield an improvement in the objective function.

It is clear from the geometry, Figure 4, that neighboring vertices have $N-1$ faces in common, hence their nonbasic sets of variables are the same except for one element. It follows that to go to a neighboring vertex, one variable must switch from the nonbasic to the basic set and one from basic to nonbasic.

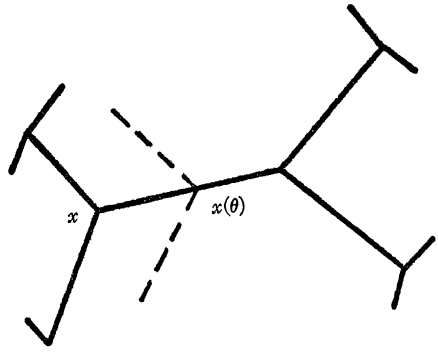

FIG. 4

Specifying the variable to enter the basic set already determines the neighboring vertex. For let us suppose that the variable y_i is to be added to the basic set. Let us give y_i the value θ and solve for the basic variables $\bar{y}$. Then $\bar{y}(\theta) = B^{-1}(b - \theta v_i)$ when v_i is the column associated with y_i. As θ increases from zero, the values of the components of $\bar{y}_i$ of $\bar{y}$ change, and there will be some last θ value at which they all are still nonnegative. Call this value $\theta_{\min}$. With $\theta = \theta_{\min}$, one of the $\bar{y}_i$, say $\bar{y}_{i_0}$, is zero. Geometrically, as θ increased, the solution point of the equations moved from the original vertex along an edge created by displacing the hyperplane corresponding to the one equation which previously had a 0 slack but now has a positive one. Finally another vertex is reached. Going beyond this point would violate the inequality represented by $\bar{y}_{i_0} \geqq 0$. Then $\bar{y}_{i_0}$ is the variable removed from the basis. The new basis consists of y_i and the remaining basic variables. With y_i at the value $\theta_{\min}$ and the remaining $\bar{y}_i$ components at their values in $\bar{y}(\theta_{\min})$ the new basic variables provide a solution to the equations. Thus we have found the coordinates of a neighboring vertex on the polyhedron.

To be able to repeat this process, we would need to find the B^{-1} corresponding to the new basic set. To do this it is not necessary to start over again. Using the old B^{-1} as a starting point, the new inverse can be obtained by a single Gaussian elimination step.

Next we turn to the question of finding a neighboring vertex that will yield an improvement. As we went up the edge from the old to the new vertex, was the objective function increasing or decreasing? Denoting by c' the row M-vector of costs corresponding to the basic variables $\bar{y}$, the change in the objective function is given by

$$c'(\bar{y}(\theta_{\min}) - \bar{y}(0)) = c'(B^{-1}(b - \theta_{\min}v_i) - B^{-1}b) = -\theta_{\min}(c'B^{-1})v_i = \theta_{\min}\pi \cdot v_i,$$

using π for the M-vector $-c'B^{-1}$.

The question of increase or decrease is determined by the sign of the scalar product $\pi \cdot v_i$. If this is positive, there is an increase, otherwise not. Note that the same π can be used in considering all possible v_i that might be introduced.

To carry out the simplex procedure, then, one starts with a vertex, and computes π. Using π, one tests for vertices that will yield an improvement by forming the scalar product with all possible columns v_i. If one of these yields a positive result, it is introduced into the basis, and one of the basic variables becomes nonbasic as described above. This process is iterated. Finally we reach a point where all the scalar products of the current π and the nonbasic vectors are zero or negative. This can easily be shown to be a maximum point.

The effectiveness of the method depends on the number of vertices that must be inspected. Each vertex requires one Gaussian elimination. Two empirically observed facts are: (i) it helps to choose as the entering column the one for which πv_i is maximal. (ii) If the rule (i) is followed, problems like (1) are usually solved in between $2M$ and $3M$ vertex inspections. Although the entire practical success of linear programming rests on observation (ii), there is, at present, no theoretical explanation.

Integer solutions. The simplex method consists of a succession of matrix inversions, the successive matrices being so related that there are great economies in passing from the inverse of one to the inverse of the next. One consequence of dealing with matrix inversions is that the calculation is not one where integral data will produce an integral answer.

Many combinatorial problems are susceptible of a formulation like equation (1), but integral answers are usually needed. For an example, consider the assignment problem. In this problem N men are to be assigned to N jobs. If the ith man is assigned to the jth job, the cost of this assignment is taken to be c_{ij}. The problem is to find that assignment which gives each man a job and each job a man, and minimizes the total cost. This can be formulated in terms of linear inequalities as follows:

$$\min \sum_{i=1}^{i=N} \sum_{j=1}^{j=N} c_{ij}x_{ij},$$

$$\sum_{j=1}^{j=N} x_{i,j} \geqq 1,\ i = 1, \cdots, N, \quad \sum_{i=1}^{i=N} x_{i,j} \leqq 1,\quad j = 1, \cdots, N \tag{12}$$

$x_{i,j} \geqq 0$, $i = 1, \cdots, N$, $j = 1, \cdots, N$. If $x_{i,j} = 1$, man i is assigned job j, if $x_{i,j} = 0$

he is not assigned job j. If the $x_{i,j}$ all turn out either 0 or 1, the inequalities in (12) will force a solution in which each man gets one assignment and each job one man. The difficulty lies in restricting the variables to be 0 or 1, and the simplex method outlined above might appear to be unable to provide an answer. A closer study of the coefficient matrix of (12) shows, however, that every square submatrix will have a determinant either $+1$, -1, or 0. Therefore, the matrix inversions involve division by ± 1 only and so will provide integral values for all the variables. In the particular case (12) it is easy to see the only possible integral values for the x_{ij} are zero and one.

A matrix all of whose sub-determinants are either $+1$, -1, or 0 is said to have the unimodular property. It is an interesting and important problem to try to discover classes of matrices having this property. A typical result which includes the matrix of (12) is

If A is the incidence matrix of the vertices versus the edges of an ordinary linear graph G, then for A to have the unimodular property it is necessary and sufficient that G have no loops with an odd number of vertices [3].

Even if a matrix does not have the unimodular property one may still want to find the best integral solution. The problem of finding the integer x that maximizes in (1) is called the integer programming problem. The integral points satisfying (1) are, of course, all within the polyhedron of feasible x's. They, together with the points that lie between them, form another polyhedron P' included in the first. All the vertices of P' are integral points. If one could produce the inequalities that yield the faces of this P', the integer problem could be solved as an ordinary linear programming problem over P'. This approach has been developed in [4] and [5] and methods of generating the faces of p' do exist. Nothing like the computational effectiveness of the ordinary simplex method has been obtained so far, however, and much work remains to be done in this area.

Nonlinear systems of inequalities. In the system (1) the linear objective function could be replaced by a nonlinear one. If the replacement is quadratic and positive definite an elegant adaptation of the simplex procedure, due to Wolfe, [6] is possible. For the general nonlinear objective function or when the inequalities involve nonlinearities, gradient methods are often used [7].

Large systems. Although the straightforward simplex method outlined above can be applied to problems of as many as a thousand inequalities and several thousand variables, many practical problems are even larger.

Usually the very large problems involve matrices with special structures. For example, there is the problem of selecting the most profitable products to produce. One unit of the jth product requires an amount a_{ij} of the ith resource. Resources are such things as labor, raw materials, etc. If the amount of the ith resource available is b_i and the profit on a unit of the jth product is c_j, then the problem is precisely (1), x_j being the amount of the jth product produced to

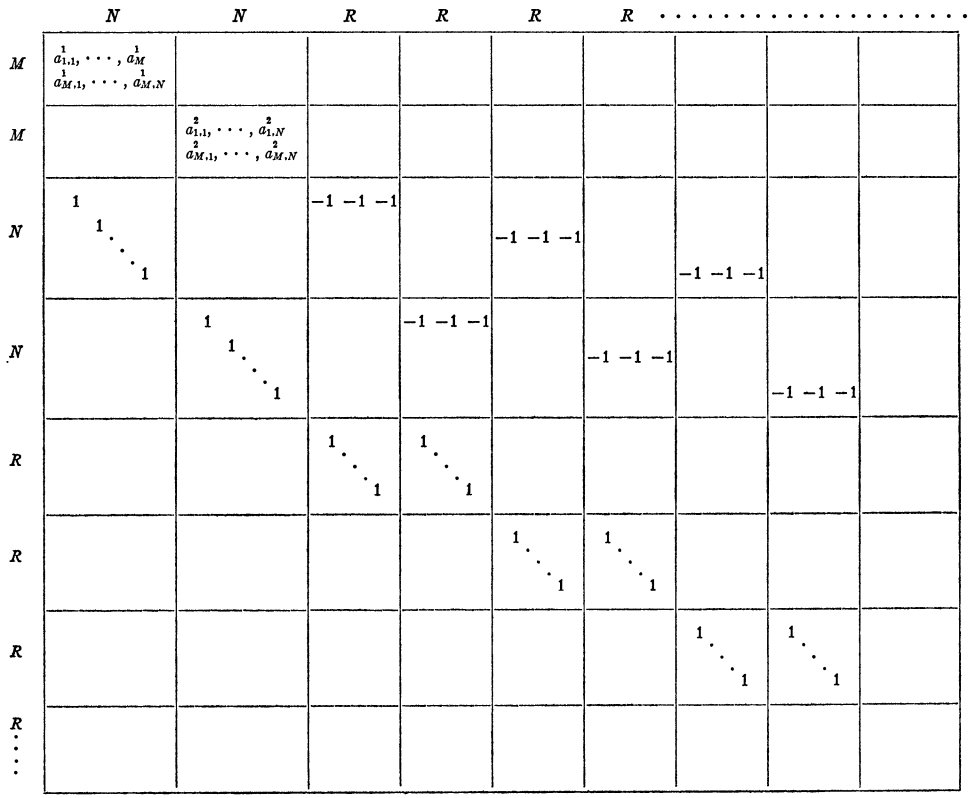

FIG. 5

maximize profit without using up more resources than are available. If we now envisage two plants and the possibility of supplying customers from either one plant or the other, depending on costs of production and transportation costs, we obtain a matrix with the structure shown in Figure 5. If either N, the number of products, or R, the number of customers, is large, the size of the matrix is largely determined by the highly structured part. Special methods for large structured problems have been developed, under the name of decomposition methods, during the past several years [8], and this is one of the most fruitful fields of current research. The spirit of these methods can be illustrated on another structured problem that leads us into the area of dynamic programming.

Imagine that identical (parent) rolls of material are to be cut up to make narrower rolls of the same diameter. We will consider all possible cutting patterns, the jth way of cutting a parent roll being described by the numbers $a_{i,j}$ of rolls of width w_i which it produces. We try to cut up the rolls to satisfy certain demands d_i for rolls of width w_i. The corresponding system of linear in-

equalities is

$$\min \sum_j x_j, \qquad \sum_j a_{ij}x_j \geqq d_i, \qquad x_j \geqq 0, \qquad i = 1, \cdots, M, \tag{13}$$

where j ranges over all possible cutting patterns and x_j is the number of parent rolls cut up using the jth pattern. One difficulty is that the x_j should be integers, but aside from this there is the fact that all the cutting patterns themselves are so numerous that this matrix could never really be written down, even for a problem in which M, the number of different widths to be produced is as small as 10 or 15.

To carry out the calculations in spite of this, [9] we start with some fairly arbitrary square submatrix A, involving M patterns, and invert that. The next stage of the calculation is where the difficulty comes in. To follow out the simplex procedure we would next form the scalar product of π with all nonbasic columns v_j and select the one maximizing $\pi \cdot v_j$ to enter the basic set. The difficulty is that there are too many columns to allow this.

This problem of finding the next column can, however, in a structured problem, be a solvable maximization problem itself. In our example (13) a column v_j is any set of nonnegative integers $y_1, \cdots, y_M$ satisfying

$$\sum_{i=1}^{i=M} y_i w_i \leqq W. \tag{14}$$

Inequality (14) simply means that the width cut out of the roll should not total more than the width W of the parent roll. Thus the problem $\max_j \pi \cdot v_j$ becomes $\max \sum_{i=1}^{i=M} \pi_i y_i$ where the y_i are nonnegative integers satisfying (13). Fortunately, this problem can be solved easily by one of the other important methods of mathematical programming, dynamic programming.

Dynamic programming.

Knapsack Problem: For the dynamic programming or recursive approach we introduce the function $v_s(x)$, which is defined to be the value of the solution to the problem

$$\max \sum_{i=1}^{i=s} \pi_i y_i, \qquad \sum_{i=1}^{i=s} w_i y_i \leqq x, \tag{15}$$

where the y_i are nonnegative integers. Problem (15) is the same as (14) but now the total width is x and only $y_1, \cdots, y_s$ can be used. $v_s(x)$ satisfies the recursion

$$v_s(x) = \max \{v_{s-1}(x), \pi_s + v_s(x - w_s)\} \tag{16}$$

since $v_s(x) = v_{s-1}(x)$ if w_s was not used in the solution to (15) (i.e. $y_s = 0$) and equals π_s plus the best use of the remaining width $x - w_s$ if it was used (i. e. $y_s \geqq 1$). Now $v_1(x)$ can be obtained trivially for all $x \leqq W$. Also $v_s(0) = 0$ for all s. Once provided with $v_1(x)$ and $v_2(0)$ we can compute $v_2(1), v_2(2), \cdots, v_2(W)$ using (16). Then with $v_2(x)$ we compute $v_3(x)$, and so on. Finally we obtain $v_M(W)$ which is the maximum under the restriction (14) and solves the original problem.

The y_i of the solution are easily obtained from (16) and form the column chosen for the next stage of the simplex method applied to the problem (13). A Gaussian elimination is then performed and the process can be iterated. In this manner the vast assemblage of possible columns or cutting patterns is never written down. Instead, the dynamic programming recursion is a device to create each column as it is needed.

Dynamic programming has a large area of applications. The example above is an illustration of its use in a combinatorial problem. The combinatorial interpretation of the problem emerges if the w_i are taken to be weights. Then we have just found the most valuable collection of objects that a man can carry in a knapsack if he is constrained by a weight limitation W and each object has a weight w_i and worth π_i. This is the origin of the name "knapsack problem."

Inventory theory. Dynamic programming can be used in what appears to be a totally different area, that of inventory theory. To see this, consider the problem of a firm wishing to keep an inventory of some item. The firm can purchase a supply y at the beginning of each week. The supply is delivered at the end of the week. During the week customers buy, depleting the inventory. The probability that the customers will want to buy an amount z during the sth week is known and is denoted by $p_s(z)$. Let us suppose that after N weeks all items remaining in inventory are disposed of at a price π per unit. The firm wishes to balance off storage costs, loss through being out of stock, etc. What amount should it buy to be most economical?

If $V_s(x)$ is the maximal expected return to the firm for starting with a supply x and maintaining inventory for s weeks with the best possible buying decisions, then $V_s(x)$ satisfies the recursion

$$V_s(x) = \max_y \left\{ -cy + \int_0^\infty [p \min(x, z) - \tfrac{1}{2}s(x + \max(0, x - z)) + V_{s-1}(\max(0, x - z) + y)]p_s(z)dz \right\}, \tag{17}$$

where c is the unit price of the items bought, s the unit storage cost, and p the sales price. Once $V_{s-1}(x)$ is known, the $V_s(x)$ values can be computed together with the y values that yield them. Since $V_0(x)$ can be taken as the value obtained by selling the remaining stock when there are 0 weeks to go, $V_0(x)$ is available, hence $V_1(x), \cdots, V_N(x)$ can be calculated using (17). If the present inventory is x, then the y that maximizes (17) in computing $V_N(x)$ is the amount to buy this week.

The recursive approach of dynamic programming is often useful in situations where a sequence of decisions must be made about a system whose state can be described by a small number of variables, and where the effect of the decision is determined by the current state alone. In the inventory example the (single)

state variable was the inventory on hand, in the knapsack example it was the amount x of unfilled space remaining.

Conclusions. A systematic discussion of dynamic programming is available in Bellman [10] and many applications are described in Bellman and Dreyfus [11]. Rather comprehensive treatments of linear programming and its applications are to be found in Dantzig [12] and Charnes and Cooper [13].

This discussion has emphasized what is known, but only a small portion of the maximization problems that one would like to solve can be attacked by the methods of either linear or dynamic programming. There are a variety of special techniques that can be used [14, 15, 16], but there are still many important maximization problems for which no computationally reasonable method is now known.

References

1. D. Gale, H. W. Kuhn, and A. S. Tucker, Linear programming and the theory of games, Chapter XIX of the Cowles Comm. for Res. in Economics Monogr. No. 13, *Activity Analysis of Production and Allocation*, Proc. Conf. on Linear Programming, Chicago, (June 20–24, 1949) Wiley, New York (1951) 317–329.

2. G. B. Dantzig, Maximization of a linear function of variables subject to linear inequalities, Chapter XXI, *ibid.* pp. 339–347.

3. A. J. Hoffman and J. B. Kruskal, Integral boundary points of convex polyhedra, Chapter in *Linear Inequalities and Related Systems*, Ann. Math. Studies No. 38, Princeton University Press, (1956) 223–246.

4. R. E. Gomory, An algorithm for integer solutions to linear programs, Chapter in *Recent Advances in Mathematical Programming*, McGraw-Hill, New York, (1963) 269–302.

5. ———, All-integer integer programming algorithm, Chapter in *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, N. J. (1963) 193–206.

6. P. Wolfe, The simplex method for quadratic programming, Econometrica, 27 (1959) 382–398.

7. W. S. Dorn, Non-linear programming, A Survey, Management Science, No. 2, 9 (June 1963) 171–208.

8. G. B. Dantzig and P. Wolfe, Decomposition principle for linear programs, Operations Res., 8 (1960) 101–111.

9. P. C. Gilmore and R. E. Gomory, A linear programming approach to the cutting stock problem—Part II, Operation Res. No. 6, 11 (1963) 863–888.

10. R. E. Bellman, Dynamic programming, Princeton University Press, 1957.

11. R. E. Bellman and S. E. Dreyfus, Applied dynamic programming, Princeton University Press, 1962.

12. G. B. Dantzig, Linear programming and extensions, Princeton University Press, 1963.

13. A. Charnes and W. W. Cooper, Management models and industrial applications, 2 vols., Wiley, New York, 1961.

14. P. C. Gilmore and R. E. Gomory, Sequencing a one-state variable machine, A solvable case of the Traveling Salesman Problem, Operations Res. No. 5, 12 (1964) 655–679.

15. J. D. C. Little, K. G. Murty, D. W. Sweeney and C. Karel, An algorithm for the Traveling Salesman Problem, Operations Res., No. 6, 11 (1963) 972–989.

16. M. Held and R. E. Karp, A dynamic programming approach to sequencing problems, J. Soc. Indust. Appl. Math., No. 1, 10 (March 1962) 196–210.