Multiattribute Decision Making by Sequential Resource Allocation

PETER A. MORRIS
Applied Decision Analysis, Inc., Menlo Park, California

SHMUEL S. OREN
Xerox Palo Alto Research Center, Palo Alto, California

(Received November 1978; accepted July 1979)

A new approach is proposed for addressing decision problems in which the outcomes are multidimensional and possibly interdependent. The method is based on decomposing the problem into a sequence of simpler decision problems. The solution to each subproblem is elicited from the decision-maker by converting it to a simple resource allocation task that may be solved by inspection. Our experience indicates that these resource allocation tasks are considerably simpler and more intuitive than the local tradeoff assessments required by existing iterative methods. The information provided by the decision-maker at each stage is used to generate a sequence of successively improving attribute bundles converging to the solution to the original complex decision problem. The approach is illustrated in the context of a financial planning problem.

THIS PAPER addresses the problem of making logical choices when there is no single measure with which to evaluate the outcomes of a decision. When the outcomes cannot be measured by a single criterion the problem of understanding and resolving the complex value issues is often the most challenging aspect of the decision. This paper introduces a methodology that helps the decision-maker by transforming the original complex assessment problem into a sequence of simpler resource allocation problems. The resource allocation problems are designed so that the preference information revealed by the decision-maker's choices is exactly the information necessary to make the original complex decision.

In order to focus on the decision-maker's preference structure we address only those problems in which uncertainty is not a dominant factor so that the decision can be modeled deterministically. There are many such decision problems in which the possible outcomes cannot be evaluated with a single criterion. Individuals, corporations and governments make daily decisions that involve different attributes such as time, money and safety.

Operations Research
Vol. 28, No. 1, January–February 1980

0030-364X/80/2801-0233 $01.25

1. BACKGROUND

To place our work in perspective it is useful to distinguish between two types of approaches to the multiattribute problem: *global modeling* and *local modeling* (Oppenheimer [9]). The global approach attempts to represent the decision-maker's preferences over all possible outcomes with a single multidimensional utility function. Typically, independence assumptions are made that reduce the function to a form having a limited number of parameters simple enough to be encoded directly. The work of Fishburn, Keelin, Keeney and Raiffa [3, 6, 7] is representative of this approach. The global approach can run into trouble when the utility function cannot be decomposed into a set of simpler lower dimensional functions. In such situations the local approach becomes relatively advantageous.

Early examples of the local modeling approach include Boyd [1] and Geoffrion, Dyer and Feinberg [5]. More recent is Dyer [2], Oppenheimer [9] and Wehrung [10]. In this approach the utility function is sampled iteratively at a set of points to create a sequence of trial solutions converging to the solution to the original decision problem. A typical scheme is based on some mathematical programming algorithm, often a variant of the Frank-Wolfe algorithm [4]. The basic idea in these algorithms is that the gradient of the objective function at each trial solution can be used to form an approximation to the objective function which, in turn, is used to generate a new trial solution. In the usual implementation the decision-maker approximates the gradient to his utility function directly by making tradeoff assessments.

A shortcoming of the local approach is that these tradeoff assessments can be very unnatural since they entail questions concerning very small changes in the attribute levels. Our approach is designed to alleviate these assessment problems by employing an algorithm specifically developed by considering the decision-maker's ability to provide information. Rather than using the decision-maker as a gradient generating subroutine in a standard mathematical programming algorithm we considered the types of questions that would be natural for a decision-maker to address. A related but quite different interactive programming method, based on similar motivations, has been described by Zionts and Wallenius [11].

2. APPROACH

The multiattribute decision problem may be represented mathematically as:

$$\max u[x(d)] \quad \text{over} \quad d \in D.$$

In this representation d is an element of the set of feasible decisions D, and x is an n-dimensional vector valued function that maps each decision

into a set of outcome measures or attributes. (These attributes are often referred to in the multiobjective optimization literature as objective functions.) Each component of x represents one of the attributes of value in the decision problem. The function u is a real valued utility function that maps the outcome space onto the real line.

In our formulation u will be implicit since it will never be necessary to specify its precise functional form. However, we shall assume the underlying preference structure is such that the indifference surfaces are convex to the origin. This implies a quasi-concave utility function whose gradient, whenever defined, always points into the positive orthant. This property is implied, for instance, by the common assumption of decreasing marginal rates of substitution. Unlike approaches based on tradeoff information we do not require differentiability.

We shall also suppress the link between the decision variables and the outcome variables so that our problem can be simplified mathematically to maximizing an implicit utility function $u(x)$ where the multiattribute outcome vector x has to satisfy an explicit set of constraints. These constraints would result from a decision model relating the set of feasible decisions to the set of possible outcomes. For simplicity we shall also assume the set of possible outcomes can be described by linear inequality constraints. Thus, our problem can be reformulated as one of maximizing an implicit utility function $u(x)$ where the n-dimensional outcome vector x has to satisfy an explicit set of constraints $Ax \leq b$ for some $m \times n$ matrix A and m-dimensional vector b. Such linear constraints can arise directly from the problem formulation or as the result of approximating a more complex outcome region. (An example of a direct linear formulation occurs when D can be modeled as a convex polyhedron in R^r and $x(d)$ is an affine mapping from D into the positive orthant of R^n. Let $x(d) = Ud + v$ where $x, v \in R^n$, U is an $n \times r$ matrix, and $D = \{d \mid Dd \leq e, x(d) \geq 0, e, d \in R^r\}$. Assuming that $r \geq n$, any outcome vector x can be obtained using a decision vector $d = (U'U)^{-1}U'(x - v)$. (For $r > n$ this decision vector is not necessarily unique.) The feasible outcome region in this case will thus be represented as $\{x \mid Ax \leq b, x \geq 0\}$ where $A = D(U'U)^{-1}U'$ and $b = e + (U'U)^{-1}U'v$. The procedure can be extended to include the decision model explicitly in cases where $x_i(d)$, $i = 1, \cdots, n$, is concave and satisfies some monotonicity properties; such generalizations are outside the scope of this paper.)

Our approach to solving the problem is based on the analogy between the multiattribute decision and the classical resource allocation problem of choosing the optimal bundle of commodities given a set of prices and a budget constraint. The attributes can be viewed as commodities while the surface defined by the budget and prices in the commodity problem is analogous to the frontier of the feasible outcome region in the decision problem.

We were motivated by the observation that in simple allocation problems it is more natural for the decision-maker to express his preference by allocating the resources directly than to provide the information required to represent his utility function mathematically. Consider, for example, the elementary problem illustrated on the left of Figure 1. (We define an elementary resource allocation problem as one in which a decision-maker must allocate a single fixed budget among commodities having fixed relative prices.) The optimal commodity allocation occurs at the point at which the indifference curves are tangent to the budget line. These indifference curves, which characterize the decision-maker's utility function, are not readily available in general. Furthermore, in this case it is simpler for the decision-maker to choose visually the most desirable point in the region bounded by his budget line than to provide the detailed information necessary to describe his utility function over the commodities.

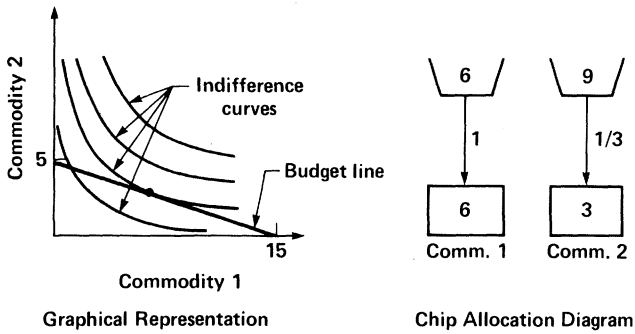

Figure 1. An elementary resource allocation problem.

An alternative way to elicit the decision-maker's preferences in this problem is shown on the right of Figure 1. Here the decision-maker must allocate a fixed set of 15 chips, representing his budget, between two buckets. The numbers in the buckets are the chips allocated to the respective commodities, which are represented in the boxes below. The arrow tags are reciprocal prices that represent the change in commodity level resulting from placing a chip in the corresponding bucket. For example, placing one chip in the second bucket will add ⅓ of a unit to the second box. This procedure facilitates the allocation process by allowing the decision-maker to move chips physically while feeding back the corresponding commodity levels, until he achieves his most preferred allocation.

While direct graphical methods are attractive in two dimensions they cannot be effectively extended to higher dimensions. The advantage of the chip allocation approach is that it can be generalized for problems

involving more than two attributes. To introduce some basic concepts we first consider the two-commodity problem illustrated on the left of Figure 2, in which the feasible region is bounded by the solid lines. One possible solution method is to convert the problem to a sequence of elementary resource allocation problems of the type described above where the budget constraint for each elementary problem corresponds to one of the line segments bounding the feasible region. If we could find such an elementary problem whose solution is in the original outcome region the solution would also be the solution to the original problem, since the feasible region is a subset of each elementary problem region. Unfortunately, this method might not work since, if the optimum is on a corner, there may be no elementary problem whose solution lies in the original outcome set. This difficulty can be eliminated by using modified elementary problems in which the outcome region is defined by cones corresponding to pairs of adjacent line segments. The shaded area in the left diagram of Figure 2 represents one such cone. Such regions still contain

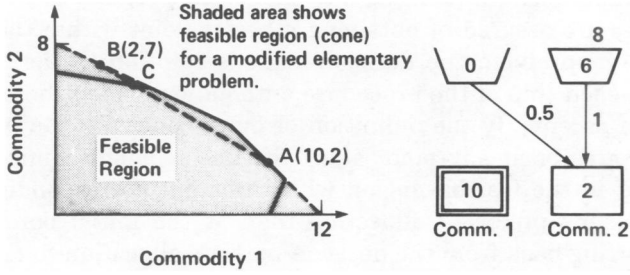

Figure 2. Modified elementary resource allocation problem.

the original outcome region; furthermore, we shall show there exists at least one cone whose solution lies in the original feasible set.

The right hand side of Figure 2 illustrates how the chip allocation scheme can be generalized to represent a cone problem. As before, the numbers on the arrows are marginal reciprocal prices that measure the impact of adding or subtracting chips from each bucket. The double lined box designates the price commodity that defines the chip units, and the lid on the right hand bucket indicates that its capacity is limited to 8 chips. Thus, the decision-maker can allocate a fixed total of 16 chips among the price commodity box and the two buckets on top. For example, moving two chips from the price commodity box to the right bucket and six more chips to the left bucket would reduce the price commodity level to 2 while increasing the other commodity level to 7. This re-allocation corresponds to moving from point A to point B on the boundary of the feasible region.

The reciprocal prices are marginal in the sense that there is no

requirement that the overall commodity levels be equal to the prices times the total chip levels (for example, in Figure 2, $(.5)(0) + (1)(6) \neq 2$). However, the prices, the initial commodity levels and the initial chip allocation are all intrinsically related through the cone problem that generated them. We shall demonstrate that placing chips in a bucket corresponds to moving within the cone region relative to one of the constraints defining its boundary, and that a lid on a bucket prevents chip allocations corresponding to points outside the cone region. Section 4 shows how the numbers in the resource allocation problem can be deduced from the cone optimization problem. The number of buckets is always less than or equal to the number of attributes regardless of the number of constraints. This is a useful fact, since in our experience most well-modeled problems do not have a large number of attributes or performance criteria even if the number of decision variables is large.

We address the problem of selecting the proper sequence of trial cones by adapting a fundamental idea in optimization theory, the concept of feasible direction. Algorithms based on this concept proceed by generating a direction of ascent pointing into the feasible region. Moving in this direction we are assured of obtaining a better point within the feasible region. In our problem the decision-maker's solution to the allocation problem at each step of the procedure automatically identifies a feasible direction of ascent. By the definition of quasi-concavity the preference function is greater at any point along the line segment connecting any initial point in the feasible region with the point corresponding to the decision-maker's preferred allocation than at the initial point. Consequently, cutting back from the decision-maker's allocation to the boundary of the feasible region produces a feasible point preferred to the initial one. Point C in Figure 2 illustrates such a cutback point obtained by backing off from point B to the boundary of the feasible region along the line segment connecting B to A. The successive points obtained at each step of our procedure are generated by cutting back the decision-maker's allocation to the boundary of the feasible region (unless the allocation itself is feasible) along the line connecting the allocation with the previous point. The successive cones defining the allocation problems are selected so each cone contains the constraints active at the previous cutback point. This ensures that the sequence of cutback points is monotonically improving with respect to the decision-maker's preference function which guarantees convergence of the procedure.

The procedure is designed to be implemented interactively on a computer. The decision-maker's role is limited to selecting his or her preferred allocation within each cone through the device of allocating scarce resource chips in a market in which attributes play the role of commodities. Each market, consisting of a set of prices, an initial attribute bundle, and a budget for the various attributes, is determined by the computer

based on the cutback procedure outlined above. The next two sections formalize the concepts introduced above to the general n-attribute case. Section 3 presents a mathematical representation of the procedure for setting up each market problem, and proves convergence of the procedure. Section 4 describes how each cone problem can be represented as a resource allocation problem.

3. THE GENERAL PROCEDURE

The basic problem under consideration, hereafter referred to as problem P, may be stated mathematically as:

$$\begin{aligned} & \max u(x) \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0. \end{aligned}$$

Recall that $u(x)$ is quasi-concave, x and b are n and m-dimensional vectors, respectively, and A is an $m \times n$ matrix. We assume the feasible set is compact. The non-negativity constraints on the attributes are for convenience only; in practice, they can be replaced by any lower bounds since the procedure below assumes the decision-maker satisfies these constraints directly by observation.

The algorithm proceeds by selecting on any iteration k a new cone C_k defined by n of the m inequality constraints (not including the non-negativity constraints) of the original problem. Such a cone is illustrated in Figure 3 and is designated by the vertex V. The constraints defining C_k include all those active at the current estimate to the solution x_{k-1} (facets I and II in Figure 3). The remaining constraints (facet III) are chosen to ensure that the vertex of the new cone is a feasible point of the original constraint set. Appendix A shows how to implement this heuristic rule that attempts to include in C_k those constraints that are closest to being active and hence are most likely to be binding in the next iteration. (Similar approaches are often used in mathematical programming to prevent zigzagging.) The cone C_k is then represented as a market problem in terms of chips and buckets, and the decision-maker selects his or her most preferred non-negative point z_k in the cone C_k via an interactive allocation process. If z_k is feasible for the original problem then it is the solution. Otherwise the line connecting x_k and x_{k-1} forms a feasible direction and the new estimate to the solution x_k is selected at the point where this line intersects the surface of the polyhedron defining the original feasible region. The above procedure assumes the number of inequality constraints m is at least as great as the number of variables n. If $m < n$ the procedure can be easily modified to produce the solution in one iteration.

The procedure is represented mathematically by the following algo-

rithm. We denote by a_i the ith row of the constraint matrix A and by b_i the corresponding element of the vector b.

Algorithm 1:

Start with an arbitrary point x_0 in the set $X \equiv \{x \mid Ax \leq b, x \geq 0\}$.

Step 1 (Cone selection)

Choose a cone C_k defined by n constraints by selecting a set of indices I_k containing n distinct integers from 1 to m such that $I_k \neq I_j$ for $j < k$, $I_k \supset \{i \mid a_i{}^T x_{k-1} = b_i\}$, and $\{x \mid a_i{}^T x = b_i, i \in I_k\} \in X$. (A method for selecting a cone satisfying these conditions is provided in Appendix A.)

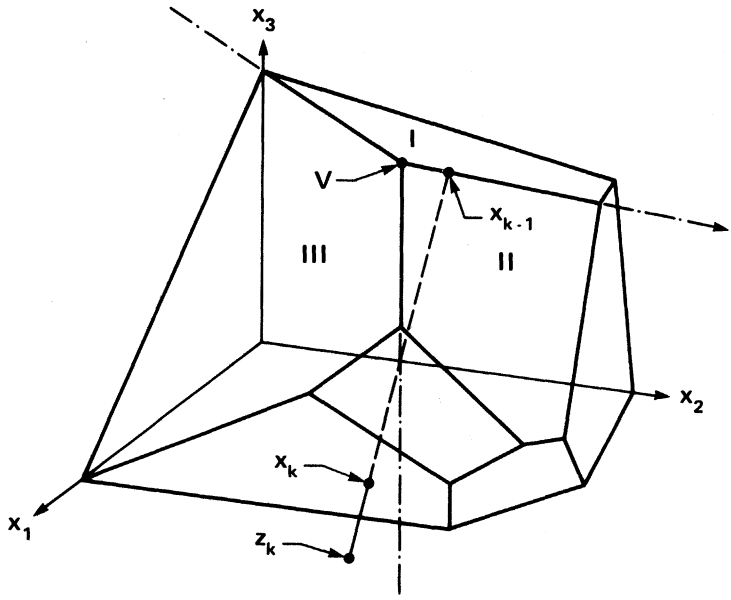

Figure 3. Illustration of the procedure.

Step 2 (Assessment)

Obtain the solution z_k to the subproblem

$$\begin{aligned} &\text{maximize} && u(z) \\ &\text{subject to} && a_i{}^T z \leq b_i \qquad \text{for} \quad i \in I_k \\ & && z \geq 0. \end{aligned}$$

In the next section we detail the procedure for converting this to a resource allocation problem.

Step 3 (Cutback)

If $z_k \in X$ then terminate (z_k is the solution).
Otherwise, set

$$x_k = x_{k-1} + \alpha_k(z_k - x_{k-1})$$

where

$$\alpha_k = \min_{i \in \{i \mid a_i{}^T(z_k - x_{k-1}) > 0,\, i \notin I_k\}} \{a_i{}^T(x_{k-1} - z_k)/(a_i{}^T x_{k-1} - b_i)\}$$

and then go to Step 1. Steps 1 and 3 of the algorithm represent tasks performed by a computer program while Step 2 is performed by the decision-maker by means of chip allocation.

Before discussing the implementation of the algorithm we examine its convergence properties. Suppose x^* is the solution to the problem. There are at most n independent active constraints at x^*. Eliminating any constraints that are not active at x^* from the original set would not have changed the solution. Thus, there exists at least one cone defined by at most n of the constraints such that the global solution x^* is the solution to the subproblem corresponding to this cone. The number of cones defined by n constraints or fewer is clearly finite, and by Step 1 of the algorithm no such cone is used more than once. Thus, a cone for which x^* is optimal will be reached within a finite number of steps which implies that the algorithm terminates in a finite number of iterations.

The finite convergence property of the above algorithm would not be as comforting if we had to perform a random search through all possible cones since the number of these cones increases exponentially with the number of inequality constraints. Fortunately, as in the Simplex Method for linear programming, we can guarantee the algorithm produces a sequence of points that monotonically improves the value of the objective function. To prove this note that z_k, the decision-maker's allocation on the kth iteration, is preferred to x_{k-1}, the initial allocation provided by the computer. Therefore, since u is quasi-concave, and $0 < \alpha_k < 1$,

$$u(x_k) = u((1 - \alpha_k)x_{k-1} + \alpha_k z_k) > \min[u(x_{k-1}), u(z_k)] = u(x_{k-1}).$$

The above property holds as long as z_k is preferred to x_{k-1}. Consequently the requirement that z_k be the optimal solution to the kth subproblem may be relaxed and replaced by a weaker requirement that z_k be preferred to x_{k-1}, and be optimal for the kth subproblem only if it is feasible for the original problem. This modification has important practical implications since it eases the assessment task considerably.

Algorithm 1 is, in a sense, more efficient than a simplex-like procedure, since it allows steps that cut across the feasible region (see Figure 3). The simplex method is restricted to moving along paths connecting adjacent vertices.

4. REPRESENTING THE SUBPROBLEM AS A CHIP ALLOCATION TASK

In this section we describe how the cone optimization subproblem defined by Step 2 of Algorithm 1 is represented as a chip allocation task for the decision-maker. Although the algebra is somewhat complex the concept is simple. The fundamental idea is to formulate the problem so each chip allocation corresponds to a different point in the outcome region. For the n-attribute problem the subproblem S under consideration has the form

$$\begin{aligned} \text{maximize} \quad & u(z) \\ \text{subject to} \quad & Bz \leq r \\ & z \geq 0, \end{aligned}$$

where B is a nonsingular $n \times n$ matrix, and r is an n-dimensional vector. By adding a vector of slack variables y we obtain the following alternative representation of the constraint set in S:

$$\begin{aligned} Bz + y &= r \\ z \geq 0, \qquad & y \geq 0. \end{aligned}$$

The point $z = B^{-1}r$, corresponding to $y = 0$, is the vertex of the cone $C = \{z \mid Bz \leq r\}$ which is feasible for problem S by construction. A positive increase in the jth component of y represents a move away from the jth constraint into the interior of C. Selecting the vector y is thus a natural mechanism for selecting a point in C. The value of z corresponding to a particular y is then calculated as $z = B^{-1}(r - y)$.

The above observation forms the basis for the procedure by which the decision-maker provides the solution to problem S. The procedure is implemented by allowing the decision-maker to select y via a chip allocation scheme that feeds back the corresponding attribute vector z. This feedback enables the decision-maker to perturb y while keeping z positive until the desired attribute bundle is achieved.

To obtain the chip allocation diagram the system is converted to the canonical form $z + B^{-1}y = B^{-1}r$. This conversion may be done at once or executed iteratively by a sequence of simplex-like pivoting steps on the tableau $[B \mid I \mid r]$ to bring it to the form $[I \mid D \mid v]$. The set of equations corresponding to the canonical tableau is then expressed as

$$z_i + y_1 d_{i1}, + y_2 d_{i2} + \cdots + y_n d_{in} = v_i \quad \text{for} \quad i = 1, 2, \cdots, n.$$

Note that the right-hand vector v in the canonical form gives the attribute vector z corresponding to the vertex of the cone C. Thus, v must be nonnegative.

The next step in creating the chip allocation scheme requires selection of one attribute, say the pth one, as a numeraire or price attribute. The corresponding pth constraint in the canonical system is then designated

as the "budget constraint." By scaling and shifting the slack variables y_i we can obtain an equivalent system in which the budget constraint contains coefficients that are all either zero or one. This is accomplished by expressing the canonical system in terms of variables z and w, where the new variable w_j $j = 1, \cdots, n$ is defined so that

$$\begin{aligned} y_j &= w_j && \text{for} \quad w_j \geq 0 && \text{if} \quad d_{pj} = 0 \\ d_{pj} y_j &= w_j && \text{for} \quad w_j \geq 0 && \text{if} \quad d_{pj} > 0 \\ d_{pj} y_j &= w_j + d_{pj} Y_j && \text{for} \quad 0 \leq w_j \leq -Y_j d_{pj} && \text{if} \quad d_{pj} < 0. \end{aligned}$$

This transformation normalizes the slack variables appearing in the budget constraint so that they are measured in terms of chips that represent units of the price attribute. The parameters Y_j in the above transformation are upper bounds on the corresponding slack variables y_j. This guarantees the feasible range of the slack variables y_j can be fully represented in terms of positive values of w_j thus avoiding the need to use negative chips. To determine Y_j we exploit the fact that by the duality theorem of linear programming (see, for example, Luenberger [8]) $y_j \leq \lambda^T r$ for any n-dimensional vector $\lambda \in \{\lambda \mid \lambda^T B \geq 0, \lambda \geq 0, \lambda_j \geq 1\}$. This suggests that if all the coefficients in the jth row of B are nonnegative we can use $Y_j = r_j$. Otherwise, Y_j can be determined by guessing a feasible λ or solving a simple linear program.

The system resulting from the above transformation will now have the form

$$z_i = -c_{i1} w_1 - c_{12} w_2, \cdots, - c_{in} w_n + u_i \quad \text{for} \quad i = 1, \cdots, n.$$

where

$$\begin{aligned} c_{ij} &= d_{ij}/d_{pj} \quad \text{if} \quad d_{ji} \neq 0 \\ &= d_{ij} \quad \text{if} \quad d_{pj} = 0 \end{aligned}$$

and

$$u_i = v_i + \textstyle\sum_{j \in \{j \mid d_{pj} < 0\}} Y_j d_{ij}$$

with w being constrained so that

$$w_j \geq 0 \quad \text{for} \quad j = 1, \cdots, n$$

and

$$w_j \leq -Y_j d_{pj} \quad \text{for} \quad j \in \{j \mid d_{pj} < 0\}.$$

The chip allocation scheme shown in Figure 4 is a direct graphical representation of the above system. The n buckets on top correspond to the n components of the vector w whose magnitudes are represented by the number of chips in each bucket. The levels of the various attributes correspond to the components of z and are specified in the n boxes on the bottom. The box distinguished by a double line corresponds to the price attribute. The boxes are linked to the buckets by arrows tagged with numbers indicating the changes in the respective attribute levels

resulting from placing a single chip in the corresponding buckets. These reciprocal prices and the direction of the arrows (an upward arrow indicates a negative reciprocal price) are obtained from the coefficients of the normalized canonical system. For example, the number on the arrow connecting box i with bucket j is the absolute value of the coefficient c_{ij}; the arrow will point toward the box if $c_{ij} > 0$ and toward the bucket if $c_{ij} < 0$. Links corresponding to zero coefficients are omitted as are links connecting the buckets with the price attribute box.

In general, there are two types of buckets. Solid-line buckets correspond to components of w that are non-zero. Lidded buckets correspond to components of w that are bounded above, and are labeled by the corresponding upper bound. The buckets designated with a broken line correspond to components of w that do not appear in the budget constraint and therefore do not affect the price attribute. Two types of chips are used in the allocation process, corresponding to the two types of buckets:

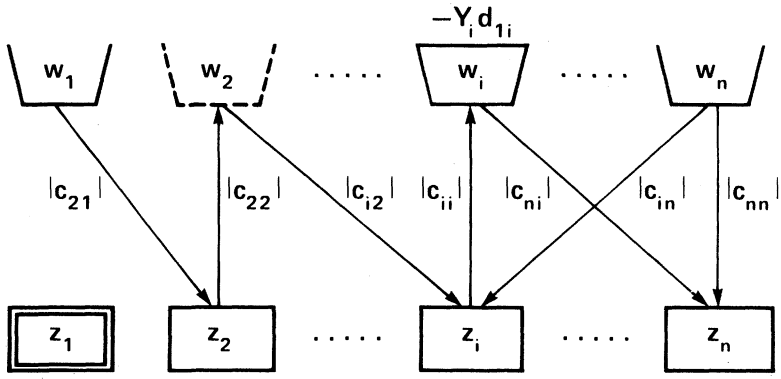

Figure 4. Chip allocation diagram.

scarce chips, each of which is worth one unit of the price attribute, and free chips. The decision-maker is given u_p scarce chips to distribute among the price attribute box and the solid-line buckets. He or she is also given an unlimited supply of free chips that may be placed only in the broken-line buckets. The free chips will not affect the price attribute, but may be used to trade other attributes among themselves.

The procedure is initialized with some arbitrary feasible distribution of chips and the corresponding attribute levels. One possibility is to start with all open buckets empty and the closed buckets filled to capacity. This is the chip allocation corresponding to the vertex of the cone C, which is feasible by assumption. The corresponding attribute levels are $z_j = v_j$ for $j = 1, \cdots, n$. Whenever a chip is placed in a bucket the attribute levels in all boxes linked to that bucket are readjusted using the system of equations relating z to w. This process is continued until the decision-maker is satisfied with the attribute levels obtained.

In summary, we have used a simplex-like tableau to facilitate the

conversion of Step 2 of the algorithm into a resource allocation problem. In fact it is possible to use an expanded version of the tableau to facilitate the implementation of the entire algorithm. The complete procedure for implementing the algorithm is presented in Appendix B. We turn now to an example application.

5. IMPLEMENTATION AND OBSERVATIONS: A FINANCIAL PLANNING EXAMPLE

To illustrate the application of the above concepts we asked a subject to solve a personal problem of balancing consumption over time by saving, borrowing and spending in a complex financial environment. We placed the subject in a hypothetical situation in which he had \$5000 in initial assets, and an income stream that increased over a four year period as follows: \$10,000, \$15,000, \$25,000, \$30,000. We created a bank at which, in any period, he could save or borrow at a 10% interest rate; however, he was restricted to borrow no more than either his total assets or 25% of his income in that period. Also, at the end of the four year period, there had to be at least \$5000 left in the bank. His asset level in each period depended on how much money he accumulated through income and investment, and on how much he decided to consume in each previous period. The subject's problem was to choose his borrowing and investment plan so as to achieve the best possible consumption pattern over the four year period. Although the financial situation was hypothetical, we asked the subject to use his true preferences in deciding among alternate consumption patterns over the four year period.

The attributes in this problem were x_i, $i = 1, \cdots, 4$, consumption in each of the four periods. The first step in our procedure was to convert the variables and constraints in the financial model into a set of constraints over the x_i's. The result of this conversion was a set of nine inequalities, corresponding to the asset and income constraints of the bank, and the \$5000 terminal constraint:

$$
\begin{array}{lll}
 & x_1 & \leq 20 \\
\text{ASSET} & 1.1\,x_1 + x_2 & \leq 36.5 \\
\text{CONSTRAINTS} & 1.21\,x_1 + 1.1\,x_2 + x_3 & \leq 64.65 \\
 & 1.33\,x_1 + 1.21\,x_2 + 1.1\,x_3 + x_4 & \leq 100.62 \\
 & x_1 & \leq 19 \\
\text{INCOME} & 1.35\,x_1 + x_2 & \leq 40.26 \\
\text{CONSTRAINTS} & 1.49\,x_1 + 1.35\,x_2 + x_3 & \leq 75.1 \\
 & 1.63\,x_1 + 1.49\,x_2 + 1.35\,x_3 + x_4 & \leq 119.3 \\
\text{TERMINAL} & & \\
\text{CONSTRAINT} & 1.33\,x_1 + 1.21\,x_2 + 1.1\,x_3 + x_4 & \leq 90.62.
\end{array}
$$

It was quickly apparent to the subject that solving the problem by observation to find the most desirable consumption pattern (x_1, x_2, x_3, x_4) was virtually impossible because of the complex interactions among all the constraints.

For convenience we implemented the procedure as an interactive computer program in which the chip allocation was displayed on a video

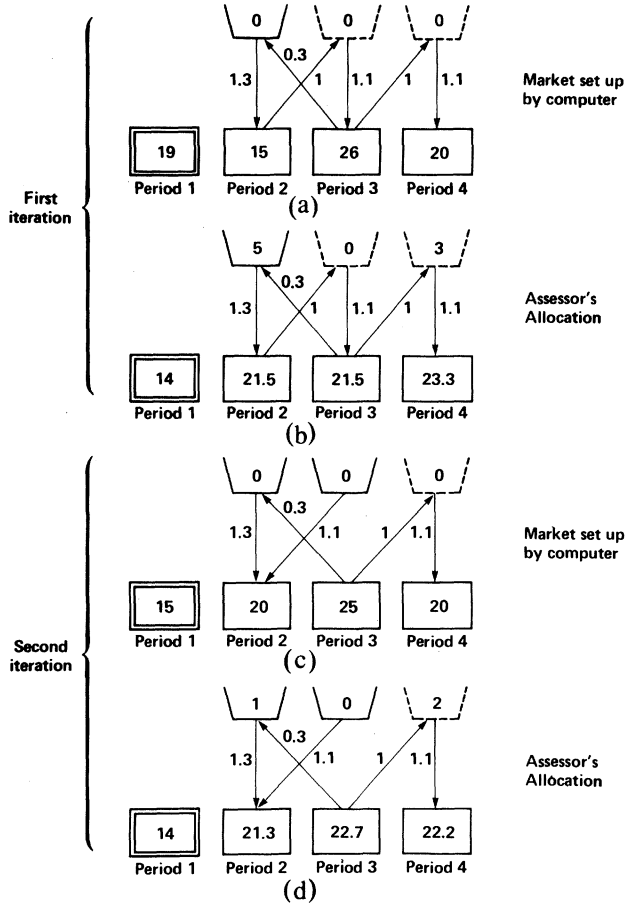

Figure 5. Illustration of the example.

screen. The program was designed so the decision-maker could move chips back and forth among the buckets on the screen while receiving instant feedback about the corresponding attribute levels in the boxes.

Figure 5a shows the initial resource allocation problem posed by the computer. The price attribute was chosen for convenience to be consumption in the first period. The subject's task was to allocate the 19

scarce chips between the first box and the solid "scarce resource" bucket and distribute as many free chips as desired between the two dashed "free resource" buckets. For the special structure of this problem it was possible to eliminate the fourth bucket since it turned out to be a free bucket with a single negative arrow connected to the period 4 box. Also, the tableau was always such that none of the buckets required lids.

The subject did not optimize his allocation in this market; we only required him to improve on the initial allocation until his allocation was outside the feasible region. His response, shown in Figure 5b, indicated a preference for relatively high uniform consumption in the latter three periods.

The computer processed the subject's allocation and then generated the new market shown in Figure 5c. In this market there were two scarce resource buckets and one free resource bucket; however, many of the market prices were unchanged so that the subject did not have to completely adjust his mind set. In response to the new market the subject chose the allocation shown in Figure 5d. He was asked to specify the best allocation possible since his most preferred point turned out to be in the original feasible region, which made it the global optimum. Thus, the procedure converged in two steps (in fact, in all our test applications so far convergence has been achieved in a small number of steps). It was simple to use the financial model to calculate the investment decisions that would enable this optimal consumption pattern.

One lesson we learned from this and other trial applications is that an essential characteristic of the assessment process is freedom for the decision-maker to experiment with different allocations. Although we have had some success with a hand calculator it is hard to give the assessor the instant feedback necessary to get a feel for the problem without access to an interactive computer with graphical capabilities. We find that people learn to allocate in this type of market system much like learning to drive a car: At first every step must be considered consciously, but later the details become second nature. We also observed that the subjects we tested have been quite comfortable with making such allocations; in fact, most have found that the process helps them think out their actual preferences.

6. CONCLUSIONS

Our research began by addressing the question "What types of information is it reasonable to expect a decision-maker to be able to provide?" The procedure we developed was based on the premise that it is natural for decision-makers to allocate resources since resource allocation is the essence of decision-making. The algorithm underlying the procedure was specifically tailored to capitalize on this human ability to provide global information, in contrast to classical mathematical programming algo-

rithms that are based on differential information about the objective function.

We believe the idea of resource allocation as a way of measuring preferences can also be useful as a general assessment technique. For example, in the global modeling approach it provides an alternative efficient way to establish the parameters of a utility function of known mathematical form. The efficiency is due to the fact that in an n-attribute problem each resource allocation has the potential of providing information equivalent to $n - 1$ tradeoff assessments.

This paper has addressed decision problems under certainty in which the feasible outcome region can be described or approximated by a set of linear inequalities. As in other iterative approaches to this type of problem, introducing uncertainty raises issues that cannot be resolved by straightforward extensions of existing methods. However, it is our hope that the concepts introduced here can be fruitfully applied to problems of decision-making under uncertainty.

APPENDIX A

Step 1 of Algorithm 1 specifies criteria for the cones used in defining the subproblem solved by the decision-maker in Step 2. These criteria require that the constraints active at the current approximation to the solution are in the set defining the next selected cone. They also require that the vertex of the cone be feasible for the original problem which ensures that it is one of the vertices adjacent to the current approximation to the solution. In general, there is no *a priori* way to identify sets of n constraints producing feasible vertices. Thus, we construct such sets iteratively using the following procedure.

Algorithm 2 (Implementing Step 1 of Algorithm 1):

At each iteration k of Algorithm 1 start with $d_0 = x_{k-1}$.

Step 1a

Select an index set $J_p \subseteq \{1, 2, \cdots, m\}$ containing n elements such that $J_p \neq I_j$ for $j < k$, and $J_p \supset \{j \mid a_j{}^T d_{p-1} = b_j\}$.

Step 1b

Obtain the point s_p satisfying the n equations:

$$a_j{}^T s_p = b_j \quad \text{for} \quad j \in J_p.$$

Step 1c

If $s_p \in X$, set $I_k = J_p$ and proceed to Step 2 of Algorithm 1. Otherwise, set

$$d_p = d_{p-1} + \beta_k(s_p - d_{p-1}),$$

where

$$p = \min_{j \in \{j \mid a_j{}^T(s_p - d_{p-1}) > 0, j \notin I_p\}} \{a_j{}^T(d_{p-1} - s_p)/(a_j{}^T d_p - b_j)\}.$$

The above procedure is analogous to the steps described in Algorithm 1. Starting with $d_0 = x_{k-1}$ we choose at any stage p of the above procedure a cone $C_k{}^p$ that is defined by n constraints containing those which are active at d_{p-1}. If the vertex of the cone, s_p, is feasible for the original problem then we let C_k be identical to $C_k{}^p$. Otherwise, we choose a new feasible point d_p at the intersection of the line connecting s_p and d_{p-1} with the feasible set X and repeat the process. Any constraint active at d_{p-1} is also active at s_p and is consequently active at d_p. Furthermore, at least one new constraint that intersects the line connecting s_p and d_{p-1} becomes active at d_p. Thus, the set of constraints active at d_p increases by at least one on each step of Algorithm 2 so the process is guaranteed to terminate in at most $n - 1$ iterations. Suppose the process terminates at iteration p^*; then d_{p^*}, which is feasible by construction, is the vertex of the resulting cone $C_k{}^{p*}$. Furthermore, the constraints defining $C_k{}^{p*}$ contain all the constraints that were active at the previous points d_p ($p = 0, 1, \cdots, p^*$) and therefore include the constraints active at x_{k-1}. The cone $C_k{}^{p*}$ thus meets the criteria given in Step 1 of Algorithm 1.

APPENDIX B

Implementation of the procedure described in the main text can be facilitated by using a simplex-like tableau to handle the bookkeeping. We begin with the initial tableau $[A \mid I \mid b]$ containing the coefficients of the equation system $Ax + y = b$. For non-negative y this system is equivalent to the inequality constraints $Ax \leq b$, and y is the corresponding "slack variable" vector. We assume without loss of generality that the $m \times n$ matrix A is such that $m \geq n$. Through a sequence of Gaussian eliminations the initial tableau can be brought to the following canonical form:

x_1	x_2	$\cdots$	x_n	y_1	y_2	$\cdots$	y_p	$\cdots$	y_{m-n}	y_{n-m+1}	$\cdots$	y_q	$\cdots$	y_m	
1	0	$\cdots$	0	0	0	$\cdots$	0	$\cdots$	0	$d_{1,m-n+1}$	$\cdots$	$d_{1,q}$	$\cdots$	$d_{1,m}$	v_1
0	1	$\cdots$	0	0	0	$\cdots$	0	$\cdots$	0	$d_{2,m-n+1}$	$\cdots$	$d_{2,q}$	$\cdots$	$d_{2,m}$	v_2
$\cdot$	$\cdot$		$\cdot$	$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$
$\cdot$	$\cdot$		$\cdot$	$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$
0	0	$\cdots$	1	0	0	$\cdots$	0	$\cdots$	0	$d_{n,m-n+1}$	$\cdots$	$d_{n,q}$	$\cdots$	$d_{n,m}$	v_n
0	0	$\cdots$	0	1	0	$\cdots$	0	$\cdots$	0	$d_{n+1,m-n+1}$	$\cdots$	$d_{n+1,q}$	$\cdots$	$d_{n+1,m}$	v_{n+1}
$\cdot$	$\cdot$		$\cdot$	$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$
$\cdot$	$\cdot$		$\cdot$	$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$
0	0	$\cdots$	0	0	0	$\cdots$	1	$\cdots$	0	$d_{p,m-n+1}$	$\cdots$	$d_{p,q}$	$\cdots$	$d_{p,m}$	v_p
$\cdot$	$\cdot$		$\cdot$	$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$
$\cdot$	$\cdot$		$\cdot$	$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$		$\cdot$		$\cdot$	$\cdot$
0	0	$\cdots$	0	0	0	$\cdots$	0	$\cdots$	1	$d_{m,m-n+1}$	$\cdots$	$d_{m,q}$	$\cdots$	$d_{m,m}$	v_m.

The basis changes in this tableau will be restricted to the slack variables y; thus, the first n columns remain unchanged throughout the procedure and may be suppressed. For simplicity assume the basic slack variables are the first $m - n$ components of the vector y and that the rows are permuted so the first m columns of the tableau form an $m \times m$ identity matrix. The actual definition is somewhat more relaxed in that a tableau is considered to be in canonical form if it can be put in the above form by permuting the columns corresponding to the slack variables.

At any stage of the algorithm the canonical tableau represents a translated cone, hereafter referred to as the *current cone*, that is defined by the inequality constraints corresponding to the nonbasic slack variables. The vertex of the current cone is given by the first n components of the right hand side: $v_1, \cdots, v_n$. This point is the value of the attribute vector x corresponding to setting the nonbasic slack variables to zero. The remaining components of the right hand side $v_{n+1}, \cdots, v_m$ give the values of the basic slack variables corresponding to the constraints not included in the current cone. Clearly, for a vector x to be feasible its components and all the corresponding slack variables must be nonnegative. Thus a cone has a feasible vertex if and only if the right hand side in its canonical tableau representation is nonnegative. Any positive increases in the values of the nonbasic slack variables represent moves away from the vertex of the current cone into its interior. For the tableau illustrated above the effect of such a move on the basic variables is given by:

$$x_i = v_i - y_{m-n+1}d_{i,m-n+1} -, \cdots, -y_m d_{i,m} \quad \text{for} \quad i = 1, \cdots, n.$$

$$y_i = v_i - y_{m-n+1}d_{i,m-n+1} -, \cdots, -y_m d_{i,m} \quad \text{for} \quad i = n + 1, \cdots, m.$$

In using the tableau format to implement Algorithms 1 and 2 we have to consider two cases.

CASE 1: (The vertex of the current cone is feasible)

This case is indicated by a nonnegative right-hand side in the tableau. The next step of the algorithm then calls for the decision-maker to select a preferred point inside the current cone. Without loss of generality we may assume the first attribute, represented by x_1, is the price attribute. Then, using the coefficients of the first row of the tableau to represent the budget constraint, we construct the chip allocation scheme as described in Section 4. The decision-maker maintains nonnegativity of z_k by inspection while the procedure automatically ensures nonnegativity of the corresponding nonbasic components of the slack variable vector y^{z_k}. If the basic components of y^{z_k} are also nonnegative then z_k is feasible, and $x_k = z_k$ is the optimal solution. If, however, at least one component of y^{z_k} is negative then z_k is not feasible and we select the new point x_k by

cutting back the step connecting the previous point x_{k-1} with z_k to the boundary of the feasible region. The point x_k is thus determined by

$$x_k = x_{k-1} + \alpha_k(z_k - x_{k-1}),$$

and the corresponding slack variable vector is

$$y_k = y_{k-1} + \alpha_k(y^{z_k} - y_{k-1}),$$

where

$$\alpha_k = \min_{i \in \{1, \cdots, m-n\}} \{ y_i^{k-1} / (y_i^{k-1} - y_i^{z_k}) \}.$$

The nonnegativity of x_k and the nonbasic components of y_k is assured by the fact that x_{k-1}, y_{k-1}, z_k and the nonbasic components of y^{z_k} are nonnegative and $\alpha_k \in [0, 1]$. Algorithms 1 and 2 require that the constraints active at x_{k-1} are in the set defining the current cone. Thus, all the zero slack variables corresponding to x_{k-1} are basic so that α_k is strictly positive. Furthermore, since we have assumed that at least one basic component of y^{z_k} is negative α_k is also strictly less than unity, implying that x_k is different from x_{k-1} or z_k.

The basic slack variables that became zero as a result of the cut back correspond to constraints active at x_k but not active at x_{k-1}. The cut back procedure guarantees existence of at least one such constraint. To maintain the canonical form of the tableau all these basic slack variables that have been driven to zero must be made nonbasic. This requires that the same number of nonbasic slack variables corresponding to inactive constraints at x_k, i.e., having strictly positive values, will be made basic. As a general rule we add to the basis slack variables having the highest values which correspond to constraints least likely to be violated on the next chip allocation step. We implement this by adding to the basis the slack variables. However, to prevent cycling we avoid choices of new basic variables that lead to a repetition of a previously considered set.

As in the simplex method the substitution of basic slack variables is performed by pivoting operations. For instance, if we want to replace in the basis the variable y_p by y_q we pivot on the element d_{pq} in the tableau which produces the new tableau coefficients as follows:

$$d'_{ij} = d_{ij} - d_{pj}\, d_{pj}/d_{pq} \quad \text{for} \quad j = 1, \cdots, m \quad \text{and all} \quad i \neq p$$

$$d'_{pj} = d_{pj}/d_{pq} \qquad \text{for} \quad j = 1, \cdots, m.$$

CASE 2: (The vertex of the current cone is infeasible)

This case occurs if at least one element on the right hand side of the tableau is negative. The procedure used in this case is somewhat analogous to the one described in Case 1. Here, however, we do not involve the decision-maker.

The vector z_k, which in Case 1 is obtained from the decision-maker's chip allocation, is selected here to be the vertex of the current cone; i.e., $z_i^k = v_i$ for $i = 1, \cdots, n$. As in Case 1 we cut back the step connecting x_{k-1} and z_k to the boundary of the feasible region. We then determine the slack variables that become nonbasic as the ones that were driven to zero by the cutback. Then we perform the appropriate pivoting to update the tableau but do not update the values of x_{k-1} and y_{k-1}. If the new right hand side of the updated tableau is still negative we repeat the above process: otherwise we execute the procedure described for Case 1.

ACKNOWLEDGMENT

This work was done while P. A. Morris was with Xerox Palo Alto Research Center.

REFERENCES

1. D. W. Boyd, *A Methodology for Analyzing Decision Problems Involving Complex Preference Assessments*, Ph.D. Dissertation, Department of Engineering-Economic Systems, Stanford University, 1970.
2. J. S. Dyer, "Time-Sharing Computer Program for the Solution of the Multiple Criteria Problem," *Management Sci.* **19,** 1379–1383 (1973).
3. P. C. Fishburn and R. L. Keeney, "Seven Independence Concepts and Continuous Multi-Attribute Utility Functions," *J. Math. Psychol.* **11,** 294–327 (1974).
4. M. Frank and P. Wolfe, "An Algorithm for Quadratic Programming," *Naval Res. Logistics Quart.* **3,** 95–110 (1956).
5. A. M. Geoffrion, J. S. Dyer and A. Feinberg, "An Interactive Approach for Multi-Criterion Optimization, with an Application to the Operation of an Academic Department," *Management Sci.* **19,** 357–368 (1972).
6. T. W. Keelin, *A Protocol and Procedure for Assessing Multi-Attribute Preference Functions*, Ph.D. dissertation, Department of Engineering-Economic Systems, Stanford University, 1976.
7. R. L. Keeney and H. Raiffa, *Decision Analysis with Multiple Conflicting Objectives: Preferences and Value Tradeoffs*, John Wiley & Sons, New York, 1976.
8. D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, Mass., 1973.
9. K. R. Oppenheimer, "A Proxy Approach to Multi-Attribute Decision Making," *Management Sci.* **24,** 675–689 (1978).
10. D. A. Wehrung, *Mathematical Programming Procedures for the Interactive Identification and Optimization of Preferences in a Multi-Attributed Decision Problem*, Ph.D. dissertation, Graduate School of Business, Stanford University, 1975.
11. S. Zionts and J. Wallenius, "An Interactive Programming Method for Solving the Multiple Criteria Problem." *Management Sci.* **22,** 652–663 (1976).