GENERATING INTERIOR SEARCH DIRECTIONS FOR MULTIOBJECTIVE LINEAR PROGRAMMING

by

Ami Arbel and Shmuel S. Oren

Memorandum No. UCB/ERL/IGCT M93/20

22 January 1993

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

GENERATING INTERIOR SEARCH DIRECTIONS FOR MULTIOBJECTIVE LINEAR PROGRAMMING

Ami Arbel
Industrial Engineering Department
Tel-Aviv University
Tel-Aviv 69978
Israel
bitnet: ami@taunos

Shmuel S. Oren
Industrial Engineering and Operations Research
University of California at Berkeley
Berkeley, CA 94720
USA
bitnet: shmuel@ucbcmsa

ABSTRACT

Currently existing multiobjective linear programming algorithms (MOLP) are using the simplex algorithm to generate a sequence of steps toward the optimal solution. The difference between the various MOLP algorithms depends on the specifics of generating these steps directions and the amount of involvement required from the decision maker (DM). This paper uses one variant of Karmarkar's interior-point linear programming algorithm and modifies it for solving multiobjective linear programming problems. Specifically, the paper considers the modification of the affine-scaling primal algorithm and develops a procedure for generating search directions that are interior to the polytope formed by the constraints of the linear programming problem. These search directions are combined to a single direction that approximates the gradient of the utility function of the decision maker at the current, interior, solution point. The solution process is comprised of a sequence of steps where search direction are generated and later combined and projected to yield the next interior iterate.

I. INTRODUCTION

The field of linear programming has long been the domain of the simplex algorithm developed by George B. Dantzig in 1947 [4,5]. This situation has changed with the introduction of a new algorithm for linear programming by Narendra K. Karmarkar in 1984 [8]. The difference between these two algorithms lies in their respective progress toward the optimal solution of the linear programming problem. While the simplex algorithm makes its progress by moving the current solution point on the *exterior* of the constraint polytope and along its vertices, Karmarkar's algorithm, in contrast, makes its progress by moving the current solution point through the *interior* of the polytope. This characteristic describes the name given to this new class of algorithms: *Interior-point methods for linear programming*. Following the original algorithm introduced by Karmarkar in 1984, a host of other variants were developed in the following years which are generally referred to as *Karmarkar-type* algorithms as all of them follow an interior trajectory toward the optimal solution. These variants now include primal, dual, primal-dual and various power-series implementations of algorithms for interior linear programming. For a representative reference list describing some of the details of these variants the reader is referred to [1,3,9,10,14].

When interior point algorithms were first introduced many claims were made in regards to their superiority relative to the simplex algorithm. With the passage of time, many performance tests were conducted (see, e.g. [6] and the references mentioned for the variants of interior point algorithms) and the general evidence points to the conclusion that, in general, the simplex algorithm outperforms interior point methods for problems of relatively small size and as the size of the problem increases, interior point methods for linear programming clearly dominate the simplex algorithm.

Current multiobjective linear programming (MOLP) algorithms are, in general, simplex-based in the sense that the solution trajectory follows a path that is exterior to the constraint polytope. Many approaches and algorithms have been developed to address the problem of vector-valued optimization problems (see, e.g. [13]). The difference between these various methods depends on the approach used in generating the search directions to guide the exterior solution path and, in addition, the amount of interaction with the decision maker (DM). As the problem grows in size, the number of vertices of the constraints polytope generally grows as well and, therefore, one can expect that a solution process that follows the vertices may become a lengthy journey. Making progress through the interior of the polytope is less sensitive to problem size and, therefore, one can expect the solution process to move faster toward a solution. Extrapolating from the experience accumulated for single-objective interior linear programming problems we can expect MOLP problems with a relatively small number of objectives and very large number of constraints to outperform simplex-based MOLP problems. Before embarking on measuring such relative performance levels, however, one has to suggest an appropriate algorithm that addresses a linear programming problem with multiple objectives while following an interior trajectory generated by an interior-point linear programming algorithm. This is precisely the purpose of this paper.

The approach developed in this paper can be viewed as an extension of the MOLP algorithm suggested in [2]. The difference between the two approaches is that in [2], all non-dominated neighboring vertices to the current one were evaluated. Next, their preference with respect to the current one were assessed and a weighted search direction, based on these preference measures, was generated to move the solution point from the current vertex to a new one. This process was repeated until no neighboring vertex was preferred to the current one. The approach developed in this paper goes through an iterative process where at each iteration we determine a single search direction corresponding to a single objective function. These search directions are then combined to provide a single search direction for the MOLP problem.

In developing an interior MOLP algorithm there are many issues to be addressed. Given the wealth of interior-point linear programming algorithms currently available, an obvious issue is the choice of algorithm to be selected for the implementation. Reviewing all algorithms is clearly beyond the scope of this paper, nevertheless, two potential candidates are offered through the so-called affine-scaling primal algorithm and the path-following primal-dual

Next we turn to discuss *scaling* of the current iterate. Given a starting vector, $\mathbf{x}=[x_1,x_2,\ldots,x_n]^T$, its components are scaled in to yield the scaled vector $\mathbf{x}_1$ whose components are of equal distance from all the walls of the polytope. Placing the scaled vector, $\mathbf{x}_1$, a *unit distance* from the walls leads us to the following scaling relationship algorithm. The choice made for this paper is to implement the affine-scaling primal algorithm in the context of a MOLP problem. While this algorithm does not have some of the performance features of other algorithms in its class, it is easy to implement and performs well for most applications.

The remainder of this paper is arranged as follows. We review the affine-scaling primal algorithm and its algorithmic details in section II. Its modification to address multiobjective linear programming problems is

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}, \qquad \mathbf{D} = \begin{bmatrix} x_1 & 0.0 & 0.0 & 0.0 \\ 0.0 & x_2 & 0.0 & 0.0 \\ & & \cdots & \\ 0.0 & 0.0 & 0.0 & x_n \end{bmatrix} \tag{10}$$

and it is easily verified that the original vector, $\mathbf{x}$, and the scaled vector, $\mathbf{x}_1$, are related through

$$\mathbf{x}_1 = \mathbf{D}^{-1}\mathbf{x}. \tag{11}$$

Furthermore, since the solution vector, $\mathbf{x}$, is always interior to the polytope (that is, $\mathbf{x}>\mathbf{0}$), the diagonal elements of the matrix $\mathbf{D}$ are strictly positive and, therefore, $\mathbf{D}$ is invertible. Scaling the original linear programming problem shown in (2) leads to the *scaled* linear programming problem given by

$$\begin{aligned} &\textbf{\textit{minimize}} && \mathbf{c}_1^T\mathbf{x}_1 \\ &\textit{subject to:} && \mathbf{A}_1\mathbf{x}_1 = \mathbf{b} \\ & && \mathbf{x}_1 \geq \mathbf{0}, \end{aligned} \tag{12}$$

where $\mathbf{x}_1 \in \mathbf{R}^n$, $\mathbf{b} \in \mathbf{R}^m$ and

$$\mathbf{A}_1 = \mathbf{AD}, \qquad \mathbf{c}_1 = \mathbf{Dc}. \tag{13}$$

The starting vector, $\mathbf{x}_1$, for this problem is placed at a unit distance from all the walls of the polytope given any starting feasible and interior vector $\mathbf{x}>\mathbf{0}$.

The projection operator for this scaled linear programming problem is given by $\mathbf{P}_1$ where

$$\mathbf{P}_1 = \mathbf{I}_n - \mathbf{A}_1^T(\mathbf{A}_1\mathbf{A}_1^T)^{-1}\mathbf{A}_1. \tag{14}$$

The descent direction vector for the scaled problem, $\mathbf{dx}_1$, is found by projecting the gradient of the scaled problem, $\mathbf{c}_1$, on the null space of $\mathbf{A}_1$ and negating its direction. This results in

$$\mathbf{dx}_1 = -\mathbf{P}_1\mathbf{c}_1 = -\mathbf{D}[\mathbf{c} - \mathbf{A}^T(\mathbf{AD}^2\mathbf{A}^T)^{-1}\mathbf{AD}^2\mathbf{c}].$$

Letting

$$\mathbf{y} = (\mathbf{AD}^2\mathbf{A}^T)^{-1}\mathbf{AD}^2\mathbf{c}, \tag{15}$$

the expression for $\mathbf{dx}_1$ simplifies to

$$\mathbf{dx}_1 = -\mathbf{D}[\, \mathbf{c} - \mathbf{A}^T\mathbf{y}\,], \tag{16}$$

and going back to the original space, the step direction vector for the new iterate, **dx**, is given by

$$\mathbf{dx} = \mathbf{Ddx}_1 = -\mathbf{D}^2[\, \mathbf{c} - \mathbf{A}^T\mathbf{y}\,]. \tag{17}$$

If we define a new variable, **z**, by $\mathbf{z}=\mathbf{c}-\mathbf{A}^T\mathbf{y}$, then the expression for the step direction vector from (17) becomes now

$$\mathbf{dx} = -\mathbf{D}^2\mathbf{z}. \tag{18}$$

When using the simplex algorithm, the expression $\mathbf{c}-\mathbf{A}^T\mathbf{y}$ represents the reduced cost vector with **y** representing the dual vector. In fact, the variable **y** defined in (15) stands for an *estimate* of that dual variable, and the vector **z**, therefore, stands for the *estimate* of the reduced cost vector. In the simplex algorithm, the next step in the iteration was determined according to *one* (negative) component of the reduced cost vector. In contrast, here the *complete* reduced cost vector, **z**, determines the next iterate.

With the step direction vector, **dx**, given by (18) we now take a step in that direction and obtain the next iterate of the solution vector, **x**. This is found from the updating formula given by

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{dx}. \tag{19}$$

Since the step direction vector, **dx**, used in this update satisfies (4), the new iterate of the solution vector, **x**, satisfies the equality constraints $\mathbf{Ax}=\mathbf{b}$. These equality constraints are satisfied regardless of how far we step along that direction. To guard against violating the *non-negativity constraints* of the solution vector, **x**, we must establish a *maximum allowable step,* α, in that direction. This modifies (19) and results in

$$\mathbf{x} = \mathbf{x}_0 + \alpha\mathbf{dx}, \qquad \alpha > 0$$

We find the parameter α by considering the non-negativity of the components of the new iterate of the solution vector:

$$x_i = x_{0_i} + \alpha dx_i \geq 0, \quad 1 \leq i \leq n \qquad \Rightarrow \qquad \alpha \geq - \frac{x_{0_i}}{dx_i},$$

where x_{0_i} and dx_i denote the i-th component of the vectors $\mathbf{x}_0$ and $\mathbf{dx}$ respectively.

Only negative components of the step direction vector, $\mathbf{dx}$, have the potential of violating the non-negativity constraints. Similar to the simplex algorithm, this maximum allowable step, α, is found from a *ratio test* shown in (20) below, where x_i is the i-th component of the current solution vector, $\mathbf{x}$, and dx_i is the i-th component of the step direction vector, $\mathbf{dx}$.

$$\alpha \equiv max \left\{ - \frac{dx_i}{x_i} \ : \ dx_i < 0, \ 1 \leq i \leq n \right\}. \tag{20}$$

Using this maximum allowable step size, while guaranteeing feasibility and maintaining the non-negativity constraints, causes at least one component of the new iterate to hit the boundary of the polytope and vanish. This causes the solution vector to no longer be interior to the feasible region. Therefore, our final modification to the update formula introduced in (19) is to multiply the step direction vector with a *stepsize* factor. With this, the new iterate of the solution vector, $\mathbf{x}$, becomes

$$\mathbf{x} = \mathbf{x}_0 + \alpha(stepsize)\mathbf{dx}, \qquad 0<stepsize<1. \tag{21}$$

While the only constraints on the stepsize are those shown in (21); values in practice, however, range from 0.95 to 0.995.

Summary of the Affine-Scaling Primal Algorithm:

Given a starting feasible and strictly interior solution vector, $\mathbf{x}_0$, to the linear programming problem in standard form, that is, $\mathbf{A}\mathbf{x}_0=\mathbf{b}$, and $\mathbf{x}_0>0$, the primal algorithm proceeds as follows:

Step 1: Set the iteration counter, k, at k=0, and initialize the solution vector through $\mathbf{x}(k)=\mathbf{x}_0$.

Step 2: Define the scaling matrix, $\mathbf{D}(k)$, as $\mathbf{D}(k)=\mathrm{diag}[x_1(k),x_2(k),\ldots,x_n(k)]$ and solve the symmetric system of equations given by

$$(\mathbf{AD}^2(k)\mathbf{A}^T)y(k)=\mathbf{AD}^2(k)\mathbf{c}$$

for the m dimensional vector $\mathbf{y}(k)$, and evaluate the reduced-cost vector, $\mathbf{z}(k)$, given by

$$\mathbf{z}(k)=\mathbf{c}-\mathbf{A}^T\mathbf{y}(k)$$

Step 3: Evaluate the step direction vector, $\mathbf{dx}(k)$, through

$$\mathbf{dx}(k)= -\mathbf{D}^2(k)\mathbf{z}(k)$$

and take a step toward the next iterate, $\mathbf{x}(k)$, that is given by

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \rho\alpha\mathbf{dx}(k)$$

where: $0<\rho<1$, and $\alpha=\max\{-dx_i(k)/x(k),\ \text{for all } dx_i(k)<0\}$

Step 4: if the problem is primal and dual feasible and the duality gap is small, STOP; otherwise, increment the iteration counter, k and goto Step 1.

With this background information we turn next to present a modified algorithm for solving multiobjective linear programming problems.

III. AN INTERIOR-POINT MULTIOBJECTIVE ALGORITHM

In this section we extend the single objective affine-scaling interior-point primal algorithm presented in the previous section to address linear programming problems with multiple objectives. We will refer to the extension of the affine-scaling algorithm to MOLP problem as the Affine-Scaling Interior MOLP (ASIMOLP) algorithm.

A multiobjective linear programming problem can be described through the following formulation

$$\begin{aligned} &\text{"max" } \mathbf{C}^T\mathbf{x} \\ &\text{subject to:} \\ &\qquad \mathbf{A}\mathbf{x}=\mathbf{b} \\ &\qquad \mathbf{x}\geq \mathbf{0} \end{aligned} \tag{22}$$

where $\mathbf{A}$ is the $m \times n$ constraint matrix, $\mathbf{C}$ is a $q \times n$ objective matrix, $\mathbf{x}$ is the n dimensional solution vector, and $\mathbf{b}$ is the m dimensional right-hand side vector of available resources. We assume that the constraint matrix, $\mathbf{A}$, is of full row rank m, and that a starting feasible and interior solution vector, $\mathbf{x}_0$, to the constraint system is available. That is, $\mathbf{A}\mathbf{x}_0=\mathbf{b}$ and $\mathbf{x}_0>\mathbf{0}$. Considering a minimization problem rather then that of maximization is easily accomplished by negating the objective functions.

The nature of the optimization problem in (22) is ambiguous since some of the objectives may be conflicting and pursuing the optimum with respect to each objective will lead to different solutions. This ambiguity is resolved by introducing the concept of a utility function $u(\mathbf{v})$ which is a real-valued function over the space of objectives attempting to capture the DM's preferences for different vectors of objective functions values as given by the objective value vector, $\mathbf{v}$. If $u(\mathbf{v})$ was explicitly available we could have used the fact that $\mathbf{v}=\mathbf{C}\mathbf{x}$ (that is, $v_i=\mathbf{c}_i^T\mathbf{x}$ where $1\leq i\leq q$) to obtain a real-valued function $u(\mathbf{x})$ and use it to arrive at the true optimal solution vector, $\mathbf{x}$, by solving the nonlinear optimization problem shown in (23).

$$\begin{aligned} &\text{"max" } u(\mathbf{x}) \\ &\text{subject to:} \\ &\qquad \mathbf{A}\mathbf{x}=\mathbf{b} \\ &\qquad \mathbf{x}\geq \mathbf{0} \end{aligned} \tag{23}$$

In general, however, $u(\mathbf{v})$ is known only implicitly by the decision maker and information about it can be elicited interactively. Since the dimension of $\mathbf{v}$ is significantly smaller than that of $\mathbf{x}$, it is cheaper to elicit preference information about $u(\mathbf{v})$ in the *objective space*, than about $u(\mathbf{x})$ in the *solution space*. Therefore, the problem we address in this paper is in the form shown in (24)

$$\begin{array}{rl} & \text{"max" } u[\mathrm{v}_1(\mathbf{x}), \mathrm{v}_2(\mathbf{x}), \ldots, \mathrm{v}_q(\mathbf{x})] \\ \text{subject to:} & \\ & \mathbf{Ax}=\mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{24}$$

and where the value of the i-th objective function is given through $\mathrm{v}_i = \mathbf{c}_i^T \mathbf{x}$.

The difficulty with this, and all other, multiobjective optimization problems is how to obtain a solution to this vector-valued optimization problem without actually evaluating the multiattribute utility function in an explicit manner. Existing solution method for MOLP problem make their progress toward a solution by considering proxy measures for this utility function without actually ever evaluating it. Such methods require interaction with the decision maker in order to adapt the progress of the solution process to reflect the decision-maker preferences. These methods, therefore, are referred to as *interactive* multiobjective optimization methods.

One may be tempted to bypass the difficulty introduced through the matrix of objective functions, $\mathbf{C}$, by trying to use some weighting scheme to reduce the q objective functions that make up the rows of the matrix $\mathbf{C}$ to a single row vector amenable to the direct application of a linear programming algorithm. The problem with such an approach is that, in general, such a weighting scheme depends on where we are in the solution space and changes from one location of the solution vector $\mathbf{x}$, to another. What we do next is to present a method that replaces the vector-valued optimization problem with a sequence of single objective optimization problems whose solution is obtained by using the affine-scaling primal algorithm.

Letting the objective matrix, $\mathbf{C}$, be written as $\mathbf{C}= [\mathbf{c1}, \mathbf{c}_2, \ldots, \mathbf{c}_q]^T$ we apply the affine-scaling primal algorithm to this problem by considering each row of the objective matrix separately. In doing so, the step direction vector, $\mathbf{dx}_i$, that maximizes the value of the i-th objective given by $\mathrm{v}_i=\mathbf{c}_i^T\mathbf{x}$, is obtained by stepping in the direction of the gradient projected on the null space of the constraints matrix $\mathbf{A}$. The next iterate, $\mathbf{x}_i$, obtained when stepping in the direction of $\mathbf{dx}_i$, is then given by

$$\mathbf{x}_i=\mathbf{x}_0+\alpha_i\mathbf{dx}_i, \tag{25}$$

where $\mathbf{x}_0$ is the, strictly interior, starting feasible solution to $\mathbf{Ax=b}$ and α_i is a step size factor chosen so that the new iterate remains interior. Note that regardless of the specific objective vector, $\mathbf{c}_i$, the projection operator, $\mathbf{P}$, used to generate the next iterate is the same for all objective vectors. As we did in the previous section, we do not evaluate the projection operator explicitly but perform its operation implicitly as follows. Incorporating the scaling operation into the generation of the step direction vector, $\mathbf{dx}_i$, we have

$$\mathbf{dx}_i= \mathbf{D}^2\mathbf{z} \tag{26}$$

where the reduced-cost vector, $\mathbf{z}$, is given through

$$\mathbf{z} = \mathbf{c}_i - \mathbf{A}^T\mathbf{y} \tag{27}$$

and where the estimate for the dual vector, $\mathbf{y}$, is found from solving the symmetric system given by

$$(\mathbf{AD}^2\mathbf{A}^T)\mathbf{y}=\mathbf{AD}^2\mathbf{c}_i \tag{28}$$

Note that introducing different objective vectors affects only the right hand side of this system of equation and the inversion (efficient solution methods actually call for factoring followed by forward and backward solve cycles; see, e.g. [7]) of $\mathbf{AD}^2\mathbf{A}^T$ is done only once for all objective vector. The approach developed in this paper solves, at each step of the algorithm, q single objective problems where the current iterate is transferred to a new one that is an ascent direction for the specific objective used to generate this step. After all new iterates are obtained, we have to combine the new iterates to a single one that improves the value of the utility function. We consider this problem next.

To solve the multiobjective optimization problem we have to find a way to approximate the utility function at the current iterate, and take a step in a direction that results in an improvement of the value of the objective function. Since the utility function depends on the objective functions, we have

$$u(v_1, v_2, \dots, v_q) = u(\mathbf{c}_1^T\mathbf{x}, \mathbf{c}_2^T\mathbf{x}, \dots, \mathbf{c}_q^T\mathbf{x})$$

and from this, the gradient with respect to the solution vector is found as

$$\nabla_x u = \frac{du}{dx} = \frac{\partial u}{\partial v_1}\frac{\partial v_1}{\partial x} + \dots + \frac{\partial u}{\partial v_q}\frac{\partial v_q}{\partial x} = \frac{\partial u}{\partial v_1}\mathbf{c}_1^T + \dots + \frac{\partial u}{\partial v_q}\mathbf{c}_q^T \tag{29}$$

which, in matrix form, is written as

$$\nabla_x u = (\nabla_v u)\mathbf{C}' \tag{30}$$

To find this approximate gradient we have to evaluate the gradient of the utility function in the objective space, that is, find $\nabla_v u$.

Considering each of the q objective function by itself, results in stepping from the initial point $\mathbf{x}_0$, along specific step direction vector, $\mathbf{dx}_i$, to q end points with their own specific values for the q objective functions. Letting Δv_{ij} denote the change in the value of the i-th objective function that results from stepping away from the initial point $\mathbf{x}_0$ along the step direction vector $\mathbf{dx}_j$. With this definition, the change in the utility function, $u(\mathbf{x})$, in stepping from the initial point $\mathbf{x}_0$ to a set of q new iterates can be approximated through a first-order Taylor's expansion as follows

$$\begin{aligned}
u(\mathbf{x}_1) &= u(\mathbf{x}_0) + \frac{\partial u}{\partial v_1}\Delta v_{11} + \frac{\partial u}{\partial v_2}\Delta v_{21} + \dots + \frac{\partial u}{\partial v_q}\Delta v_{q1} \\
u(\mathbf{x}_2) &= u(\mathbf{x}_0) + \frac{\partial u}{\partial v_1}\Delta v_{12} + \frac{\partial u}{\partial v_2}\Delta v_{22} + \dots + \frac{\partial u}{\partial v_q}\Delta v_{q2} \\
&\dots \\
u(\mathbf{x}_q) &= u(\mathbf{x}_0) + \frac{\partial u}{\partial v_1}\Delta v_{1q} + \frac{\partial u}{\partial v_2}\Delta v_{2q} + \dots + \frac{\partial u}{\partial v_q}\Delta v_{qq}
\end{aligned} \tag{31}$$

In matrix form, these q approximations are written as

$$\Delta \boldsymbol{u} = \Delta[u(\mathbf{x})] = (\nabla_v u)(\Delta \mathbf{V}) \tag{32}$$

where the i-th component of the vector Δu is given by $u(\mathbf{x}_i)$-$u(\mathbf{x}_0)$, the q-dimensional row vector $\nabla_v u$ is the gradient of the utility function given by

$$\nabla_v u = \left[\frac{\partial u}{\partial v_1} \; \frac{\partial u}{\partial v_2} \; \cdots \; \frac{\partial u}{\partial v_q} \right] \tag{33}$$

and the q ×q matrix $\Delta \mathbf{V}$ is given by

$$\Delta \mathbf{V} = \begin{bmatrix} \Delta v_{11} & \Delta v_{12} & \cdots & \Delta v_{1q} \\ \Delta v_{21} & \Delta v_{22} & \cdots & \Delta v_{2q} \\ & & \cdots & \\ \Delta v_{q1} & \Delta v_{q2} & \cdots & \Delta v_{qq} \end{bmatrix} \tag{34}$$

From this approximation, the gradient of the utility function with respect to the objective vector, $\mathbf{v}$, is given through

$$\nabla_v u = \Delta u (\Delta \mathbf{V})^{-1} \tag{35}$$

Since the changes in the values of the individual objective functions are given by

$$\Delta v_{ij} = \mathbf{c}_1^T(\mathbf{x}_0 + \alpha_j \mathbf{dx}_j) - \mathbf{c}_1^T \mathbf{x}_0 = \alpha_j \mathbf{c}_1^T \mathbf{dx}_j \tag{36}$$

we can express the q×q matrix $\Delta \mathbf{V}$ as

$$\Delta \mathbf{V} = \mathbf{C}[\mathbf{dx}_1, \mathbf{dx}_2, \ldots, \mathbf{dx}_q] \begin{bmatrix} \alpha_1 & & \\ & \cdots & \\ & & \alpha_q \end{bmatrix} = \mathbf{C}(\mathbf{DPD})\mathbf{C}^T \begin{bmatrix} \alpha_1 & & \\ & \cdots & \\ & & \alpha_q \end{bmatrix} \tag{37}$$

where $\mathbf{P}=\mathbf{I}_n-\mathbf{A}_1^T(\mathbf{A}_1\mathbf{A}_1^T)^{-1}\mathbf{A}_1$ is the projection operator for the null space of the scaled matrix $\mathbf{A}_1=\mathbf{AD}$ which we evaluate indirectly by solving the symmetric system of equations shown in the previous section.

Observing the result for the approximated gradient of the utility function, we see that as long as we remain interior, the scaling matrix $\mathbf{D}$ has a strictly positive diagonal elements. Furthermore, if no row of the objective matrix $\mathbf{C}$ is in the range space of the constraint matrix $\mathbf{A}$, that is, no row of $\mathbf{C}$ can be expressed as a linear combination of the rows of the constraints matrix **A**, matrix $\Delta\mathbf{V}$ is invertible and, therefore, we can solve the system of equations for the approximated gradient of the utility function.

With the value of the gradient of the utility function is objective space given, we now return to the expression of the gradient in the solution space. Doing so, results in

$$\nabla_{\mathbf{x}} u = (\nabla_{\mathbf{v}} u)\mathbf{C} = \Delta \boldsymbol{u}(\Delta\mathbf{V})^{-1}\mathbf{C} \tag{38}$$

The Taylor's series approximation for the utility function involves the value of the utility function at the initial point $\mathbf{x}_0$ as well as the value at the new iterates. In the absence of the true utility function, these values are unavailable and have to be expressed through some proxy measures. There is no single way for doing it and we choose here to use the Analytic Hierarchy Process (AHP) to assess priorities associated with the utility of the initial solution point $\mathbf{x}_0$ as well as the q new iterates, $\mathbf{x}_1,\ldots,\mathbf{x}_q$. Since the basic operations involved in deriving priorities using the AHP are well known by now, we do not dwell on the specific details. The reader is referred to [11,12] for details of the AHP.

To obtain an approximate measure for the utility function at the $q+1$ points of interest we proceed as follows. While the value of the utility function at the initial point x_0 and the q new points $x_1,x_2,\ldots,x_q$ is unknown, we can still evaluate the complete q-dimensional vector of objective function values, $\mathbf{v}(\mathbf{x})=\mathbf{C}\mathbf{x}$ at each of these points. Together with the current iterate, $\mathbf{x}_0$, we have information about $q+1$ points in objective space which we present to the DM in order to obtain a measure of relative preference for these end points. This step is done by using the AHP and involves filling a comparison matrix that evaluates the relative preference of these points.

After the comparison matrix is filled, its principal eigenvector provides the priority vector that is used to provide an approximate measure of the vector $\Delta \boldsymbol{u}$. The relation between the priority vector and the utility values along the different directions is through the component-wise relation given by

$$\frac{u_i}{u_j} = \frac{p_i}{p_j} \qquad \text{for all } 0 \le i \le q, \quad \text{and } 0 \le j \le q$$

This results in the row vector, Δu being approximated through

$$\Delta u = \beta \Delta \mathbf{p} = \beta[\, p_1 - p_0,\ p_2 - p_0,\ \dots,\ p_q - p_0\,] \tag{39}$$

for some unknown scalar β and where p_i (i=0,1,...,q) is the priority of the i-th iterate as derived by using the AHP. With this vector now available, we have all the requirements for evaluating the gradient of the utility function with respect to $\mathbf{v}$ as shown in (38).

It is useful to seek an interpretation for this approximate gradient and its relation to the priorities established through the comparison matrix of the AHP. We established earlier the following result

$$\nabla_v u = \Delta u (\Delta \mathbf{V})^{-1} \quad \Rightarrow \quad \Delta u = (\nabla_v u)(\Delta \mathbf{V}) \tag{41}$$

Using the expression for $\Delta \mathbf{V}$ as given in (37), the relation between the approximate gradients $\nabla_x u$ and $\nabla_v u$ as given by (38), and defining a diagonal $q \times q$ coefficients matrix, Ω, through

$$\Omega = \begin{bmatrix} \alpha_1 & & \\ & \cdots & \\ & & \alpha_q \end{bmatrix}$$

we derive the following expression for the vector Δu

$$\begin{aligned} \Delta u &= (\nabla_v u)\mathbf{C}[\mathbf{dx}_1, \mathbf{dx}_2, \dots, \mathbf{dx}_q]\Omega = \\ &= (\nabla_x u)[\mathbf{dx}_1, \mathbf{dx}_2, \dots, \mathbf{dx}_q]\Omega = \\ &= [\alpha_1 < \nabla_x u, \mathbf{dx}_1 > , \dots , \alpha_q < \nabla_x u, \mathbf{dx}_q >] \end{aligned} \tag{42}$$

where $<\nabla_x u, \mathbf{dx}_i>$ designates the inner product between the gradient vector $\nabla_x u$ and the i-th step direction vector $\mathbf{dx}_i$.

With the approximate gradient now available, we have to generate the next iterate and move from our current position given by $\mathbf{x}_0$ to a new iterate. Using the approximation for the utility function through the approximate gradient we have

$$u(\mathbf{x}) \approx u(\mathbf{x}_0) + (\nabla_x u)\mathbf{dx} \tag{43}$$

where $\mathbf{dx}$ is the step direction vector to be used in making the actual next step. Clearly this new step direction vector is some combination of the set of individual step direction vectors $\{\mathbf{dx}_i\}$ established for each of the q single objective linear programming problems.

The approximate problem for determining $\mathbf{dx}$ is given through

$$\begin{aligned} &\text{"max" } [\, u(\mathbf{x}_0) + (\nabla_x u)\mathbf{dx} \,] \\ &\text{subject to:} \\ &\qquad \mathbf{A}(\mathbf{x}_0 + \mathbf{dx}) = \mathbf{b} \\ &\qquad \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{44}$$

and since the current iterate, $\mathbf{x}_0$, is feasible, this is equivalent to

$$\begin{aligned} &\text{"max" } (\nabla_x u)\mathbf{dx} \\ &\text{subject to:} \\ &\qquad \mathbf{A}\mathbf{dx} = \mathbf{0} \\ &\qquad \mathbf{x}_0 + \mathbf{dx} \geq \mathbf{0} \end{aligned} \tag{45}$$

If we use the affine-scaling primal algorithm to find $\mathbf{dx}$, we project $\nabla_x u$ instead of $\mathbf{c}$. That is, remembering the scaling operation, we have

$$\mathbf{dx} = \mathbf{DP}(\mathbf{D}\nabla_x u)^T \tag{46}$$

where $\mathbf{P} = \mathbf{I}_n - \mathbf{DA}^T(\mathbf{AD}^2\mathbf{A}^T)^{-1}\mathbf{AD}$ is the same projection operator used in finding each of the individual step direction vectors $\{\mathbf{dx}_i\}$, where $\mathbf{dx}_i = \mathbf{D}[\mathbf{P}(\mathbf{Dc}_i)]$. Since, from (38), $\nabla_x u$ is a linear combination of the rows of $\mathbf{C}$, the step direction vector $\mathbf{dx}$ must be a linear combination of the individual step direction vectors $\{\mathbf{dx}_i\}$. To show it we proceed as follows. Let $\mathbf{D}_x$ be the $n \times q$ matrix whose columns are the individual step direction vectors, that is $\mathbf{D}_x = [\mathbf{dx}_1, \mathbf{dx}_2, \ldots, \mathbf{dx}_q] = \mathbf{DPDC}^T$, and let Ω be the $q \times q$ diagonal matrix given by $\Omega = \mathrm{diag}(\alpha_1, \ldots, \alpha_q)$. Then, if $\mathbf{w} \in \mathbf{R}^q$ is some vector of weights, we have

$$\mathbf{dx} = (\mathbf{D}_x)\mathbf{w} \tag{47}$$

and therefore

$$(\mathbf{D}_x)\mathbf{w} = \mathbf{DP}(\mathbf{D}\nabla_x u)^T \tag{48}$$

Multiplying both sides of (48) by $(\mathbf{D_x})^T\mathbf{D}^{-2}$ we have

$$(\mathbf{D_x})\mathbf{D}^{-2}(\mathbf{D_x})\mathbf{w} = (\mathbf{D_x})\mathbf{D}^{-2}\mathbf{DP}(\mathbf{D}\nabla_{\mathbf{x}}u)^T \tag{48}$$

which leads to

$$(\mathbf{CDPDC}^T)\mathbf{w} = (\mathbf{CDPD})(\nabla_{\mathbf{x}}u)^T \tag{49}$$

To proceed, we note that since - from (37) - $\Delta\mathbf{V}=\mathbf{CDPDC}^T\Omega$, and from (38)-(39) we have $\nabla_{\mathbf{x}}u=\beta\Delta u(\Delta\mathbf{V})^{-1}\mathbf{C}$, we can rewrite (49) as

$$(\Delta\mathbf{V}\Omega^{-1})\mathbf{w} = \beta(\mathbf{CDPD})\mathbf{C}^T[(\Delta\mathbf{V})^{-1}]^T\Delta p^T \tag{50}$$

from which we have $\mathbf{w}$ explicitly through

$$\mathbf{w} = \beta[(\Delta\mathbf{V})^{-1}]^T\Delta p^T \tag{51}$$

With this weight vector, the step direction vector, $\mathbf{dx}$, is given through

$$\mathbf{dx} = \beta\mathbf{D_x}[(\Delta\mathbf{V})^{-1}]^T\Delta\mathbf{p}^T \tag{52}$$

and, therefore,

$$\mathbf{dx}^T = \Delta u(\Delta\mathbf{V})^{-1}(\mathbf{D_x})^T \tag{53}$$

which is similar to the linear combination we had in (38) but with the columns of $\mathbf{D_x}$ rather than the rows of $\mathbf{C}$.

With the combined step vector, $\mathbf{dx}$, now available, we perform the ratio test to determine the maximum allowable step size and taking a certain fraction of it (to remain interior) we arrive at the next iterate.

The number of step direction vectors generated at each step equals the number of objective vectors. Starting with the second iteration, however, we have one additional vector along which we can take our next step. Recall that the new iterate, $\mathbf{x}$, was obtained from

$$\mathbf{x} = \mathbf{x}_0 + \rho\alpha\mathbf{dx}$$

where α is determined from the minimum ratio test, and ρ is a step size factor satisfying $0<\rho<1$ that keeps the next iterate interior to the polytope. By allowing ρ=1 we take the next step all the way to the boundary to an *end point* that is given through

$$\mathbf{x}_{\text{end}} = \mathbf{x}_0 + \alpha\mathbf{dx} \tag{54}$$

The *maximal step direction vector*, $\mathbf{dx}_{\text{end}}$, that takes us from the current position, , to the position given by $\mathbf{x}_{\text{end}}$, is easily evaluated through

$$\mathbf{dx}_{\text{end}} = \mathbf{x}_{\text{end}} - \mathbf{x}_0 \tag{55}$$

Now, in addition to the q step direction vectors evaluated through the affine-scaling primal algorithm we also have this direction as a candidate along which to continue our steps. This end point also serves as an *anchor point* that one can jump to if the new directions generated at a given iteration lead to less desirable points than the current one.

The process continues until we either terminate at the wall or when the new iterates are less preferred than the current one. In this case we simply move the current iterate along the last step direction vector and take it all the way to the boundary of the polytope.

We summarize the proposed algorithm below.

An Affine Scaling Interior Multiobjective Linear Programming Algorithm (ASIMOLP):

Step 0: Given a starting feasible and interior solution, $\mathbf{x}_0$, set iteration counter k=0, set current iterate $\mathbf{x}$=$\mathbf{x}_0$,

Step 1: Increment iteration counter k:=k+1. Define the scaling matrix $\mathbf{D}$ through $\mathbf{D}$=diag[$x_1, x_2, \ldots, x_2$] where x_i is the i-th component of the current solution vector $\mathbf{x}$, and solve for the set of single step direction vectors {$\mathbf{dx}_i$} from

$$(\mathbf{AD}^2\mathbf{A}^T)\mathbf{y}_i = \mathbf{AD}^2\mathbf{c}_i \qquad i=1,2,\ldots,q$$
$$\mathbf{z}_i = \mathbf{c}_i - \mathbf{A}^T\mathbf{y}_i$$
$$\mathbf{dx}_i = \mathbf{D}^2\mathbf{z}_i$$

Next find the set of maximum allowable step sizes $\{\alpha_i\}$ by performing the ratio test for each of the step direction vectors $\mathbf{dx}_i$, and find the set of new iterates $\{\mathbf{x}_i\}$ by using

$\mathbf{x}_i = \mathbf{x} + \rho\alpha_i\mathbf{dx}_i$ where ρ is a stepsize factor satisfying $0<\rho<1$.

Step 2: Using the AHP, construct a comparison matrix. If this is the first iteration, we construct a $(q+1)\times(q+1)$ matrix and find the priority vector associated with each of the new iterates $\{\mathbf{x}_i\}$ as well as that associated with the current iterate $\mathbf{x}_0$. Denoting this vector of priorities by $\mathbf{p}$, find the q-dimensional vector Δu through

$\Delta \boldsymbol{u} = \beta\Delta\mathbf{p} = \beta[p_1\text{-}p_0, p_2\text{-}p_0, \dots, p_q\text{-}p_0]$ where β is yet unknown and p_i is the i-th component of the $q+1$ dimensional vector $\mathbf{p}$.

for any iteration after the first, we also consider the value at the boundary point, $\mathbf{x}_{end}$, (which we derive at step 5). In that case, we have a $(q+2)\times(q+2)$ comparison matrix and the resulting vector of priorities is of dimension $q+2$.

Step 3: Evaluate the value matrix, $\Delta\mathbf{V}$ as follows. If this is the first iteration, that is, k=1, the matrix is given through

$$\Delta\mathbf{V} = \mathbf{C}[\mathbf{dx}_1, \mathbf{dx}_2, \dots, \mathbf{dx}_q]\begin{bmatrix} \alpha_1 & & \\ & \cdots & \\ & & \alpha_q \end{bmatrix}$$

otherwise, for k>1 we have

$$\Delta\mathbf{V} = \mathbf{C}[\mathbf{dx}_1, \mathbf{dx}_2, \dots, \mathbf{dx}_q, \mathbf{dx}_{end}]\begin{bmatrix} \alpha_1 & & & \\ & \cdots & & \\ & & \alpha_q & \\ & & & \alpha_{end} \end{bmatrix}$$

Next, find the approximate gradient $\nabla_x u$ according to

$$\nabla_x u = (\nabla_v u)\mathbf{C}$$

Step 4: Find the step direction vector $\mathbf{dx}$ and the new iterate, $\mathbf{x}$, by solving

$$(\mathbf{AD}^2\mathbf{A}^T)\mathbf{y} = \mathbf{AD}^2(\nabla_x u)^T$$
$$\mathbf{z} = (\nabla_x u)^T - \mathbf{A}^T\mathbf{y}$$
$$\mathbf{dx} = \mathbf{D}^2\mathbf{z}$$

Find the maximum allowable step size α by performing the ratio test and find the set of new iterates $\mathbf{x}$ through

$$\mathbf{x} = \mathbf{x} + \rho\alpha\mathbf{dx}$$

where ρ is a stepsize factor satisfying $0<\rho<1$.

Step 5: Find the boundary point, $\mathbf{x}_{end}$, as follows. If this is the first iteration, that is, k=1, the boundary point, $\mathbf{x}_{end}$, is obtained by taking the largest step along the direction vector $\mathbf{dx}$

$$\mathbf{x}_{end} = \mathbf{x} + \alpha\mathbf{dx}.$$

for any iteration other than the first, that is $k\geq 2$, we first evaluate a *candidate*, $\bar{\mathbf{x}}$, for a new boundary point through

$$\bar{\mathbf{x}} = \mathbf{x} + \alpha\mathbf{dx}.$$

If the value vector at this point $\bar{\mathbf{v}}=\mathbf{C}\bar{\mathbf{x}}$ is preferred to that available at $\mathbf{x}_{end}$ we replace the current boundary point with the new point, that is $\mathbf{x}_{end}=\bar{\mathbf{x}}$. With the boundary point now available, the vector $\mathbf{dx}_{end}$, used in step 3, is found from

$$\mathbf{dx}_{end} = \mathbf{x}_{end} - \mathbf{x}.$$

Step 6: If at step 2, all the components of the vector Δu are negative, no new direction is preferred to the current one. In this case set $\mathbf{x}=\mathbf{x}_{end}$ and terminate the iteration process, OTHERWISE: goto step 1.

Remarks:

1. At step 1, the i-th new end point is found from $\mathbf{x}_i = \mathbf{x} + \rho\alpha_i\mathbf{dx}_i$ where ρ is a stepsize factor satisfying $0<\rho<1$. Before presenting it to the

decision maker, a dominance check can be performed relative to the current iterate, $\mathbf{x}_0$. It the new point is dominated it is advisable to limit the actual step taken along this direction. This can be accomplished by using a much smaller stepsize factor ρ. The reason is that since the priorities provide an approximation to the utility function, a smaller step will provide a better approximation. Large steps should only be taken in non-dominated directions.

2. In step 4, note that by determining the step size directly through the ratio test that takes us all the way to the boundary, we can eliminate the unknown constant β.

3. In step 5, preference of the value vector at $\mathbf{x}_{end}$ relative to that at $\bar{\mathbf{x}}$ is evaluated directly using the AHP. If the utility function is available, preference is determined from comparing $u(\bar{\mathbf{x}})$ and $u(\mathbf{x}_{end})$. The purpose of this step is to keep updating the boundary point during the interactive process and to keep the best one. This allows us to have one additional direction vector to consider in evaluating the approximate gradients, as well as providing us with a solution point at which we can terminate the search for an optimal solution.

Next, we demonstrate the procedure outlined above with a simple numerical example.

IV. A NUMERICAL EXAMPLE

Consider the problem given by

$$\text{"max"} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}$$

subject to:

$$x_1 + x_2 \leq 10$$

$$\mathbf{x} \geq \mathbf{0}$$

with an initial interior feasible solution given by $\mathbf{x}_0^T=[2,1,7]$. To validate the proposed approach it is useful to use an actual utility function in generating the iterative process rather then using the AHP procedure. Assuming that the utility function is given by $u(\mathbf{x})=u(x_1 x_2)=x_1 x_2$, the true optimal solution that maximizes the value of this utility function is at the point where $x_1=x_2=5$, with the optimal value for the utility function given by $u^*=25$. Next we go through one step of the modified affine-scaling primal algorithm.

Using the affine scaling primal algorithm, the two step direction vectors, $\mathbf{dx}_1$ and $\mathbf{dx}_2$, are given by

$$\mathbf{dx}_1=\begin{bmatrix} 3.7037 \\ -0.0741 \\ -3.6296 \end{bmatrix} \qquad \mathbf{dx}_2=\begin{bmatrix} -0.0741 \\ 0.9815 \\ -0.9074 \end{bmatrix}$$

performing the respective ratio tests, and taking a stepsize factor of 0.15, the two new iterates are given by

$$\mathbf{x}_1=\begin{bmatrix} 3.0714 \\ 0.9786 \\ 5.9500 \end{bmatrix} \qquad \mathbf{x}_2=\begin{bmatrix} 1.9143 \\ 2.1357 \\ 5.9500 \end{bmatrix}$$

Since the utility function is given, we evaluate the vector $\Delta \boldsymbol{u}$ directly

$$\Delta \boldsymbol{u} = [\ 1.0056 \quad 2.0884\ ]$$

Note that we ignore the proportionality constant β since, as discussed earlier, we incorporate it when determining the step factor α.

With the change in values of the objective functions is summarized in the matrix $\Delta\mathbf{V}$ given by

$$\Delta\mathbf{V} = \begin{bmatrix} 1.0714 & -0.0857 \\ -0.0214 & 1.1357 \end{bmatrix}$$

the approximate gradients are given by

$$\nabla_v u = [0.9768 \quad 1.9125]$$

$$\nabla_x u = [0.9768 \quad 1.9125 \quad 0.0000]$$

The new step direction vector, **dx**, the new iterate, **x**, and the first boundary point which is the maximal new iterate, $\mathbf{x}_{end}$, are given now by

$$\mathbf{dx}=\begin{bmatrix} 3.4762 \\ 1.8048 \\ -5.2810 \end{bmatrix} \qquad \mathbf{x}=\begin{bmatrix} 2.2304 \\ 1.1196 \\ 6.6500 \end{bmatrix} \qquad \mathbf{x}_{end}=\begin{bmatrix} 6.6078 \\ 3.3922 \\ 0.0000 \end{bmatrix}$$

and the maximal step direction vector, $\mathbf{dx}_{end}$ is found from $\mathbf{x}_{end}$-$\mathbf{x}$. Summary of the first 25 iterations are shown in the table below. It is useful to note that both **x** and $\mathbf{x}_{end}$ converge to the true optimal solution.

Table 1, Summary of iterations

	x			$\mathbf{x}_{end}$		
Iteration	x_1	x_2	x_3	x_1	x_2	x_3
1	2.2304	1.1196	6.6500	6.6078	3.3922	0.0000
2	2.9007	1.4468	5.6525	6.6078	3.3922	0.0000
3	3.4542	1.7412	4.8046	6.5903	3.4097	0.0000
4	3.9004	2.0157	4.0839	6.4287	3.5713	0.0000
5	4.2481	2.2806	3.4713	6.2184	3.7816	0.0000
6	4.4908	2.5586	2.9506	5.8664	4.1336	0.0000
7	4.6238	2.8682	2.5080	5.3774	4.6226	0.0000
8	4.7899	3.0782	2.1318	5.3774	4.6226	0.0000
9	4.7682	3.4197	1.8121	4.6452	5.3548	0.0000
10	4.7075	3.7522	1.5403	4.6452	5.3548	0.0000
...						
15	4.9531	4.3635	0.6834	4.7505	5.2495	0.0000
...						
20	4.8486	4.8481	0.3032	5.0033	4.9967	0.0000
...						
25	4.9187	4.9486	0.1345	5.0033	4.9967	0.0000

The trace of the interior solution trajectory is shown in Fig. 1.

Fig. 1, The solution trajectory

The evolution of the value of the utility function both at the current iterate as well as at the current end point is shown in Fig. 2. Note that a good candidate for a termination point is obtained rather quickly in the process.

Fig. 2, The utility value along the solution trajectory

V. SUMMARY

We developed in this paper a modification of an interior point linear programming algorithm suitable for problems having multiple objectives. The proposed algorithm is based on modifying the affine scaling primal algorithm to yield an approximate gradient that is than projected to yield the step direction in which to take the next step. While the basic mechanism for implementing such an algorithm have been developed and presented in this paper, there are more issues that are worthwhile to look at in future work. An obvious open question at this point is that of selecting the stepsize factor used in taking the interior steps. In this paper we used a constant fraction but it is of interest to explore the effect of a variable stepsize in the iteration process. Another issue worthy of further study is that of the termination condition. Also of interest is the choice of interior point linear algorithm to be selected for a MOLP implementation. Finally, once a robust implementation is available, it will be of interest to compare it against simplex-based MOLP algorithms and observe performance differences.

REFERENCES

1. I. Adler, M.G.C. Resende, G. Veiga and N.K. Karmarkar, "An implementation of Karmarkar's algorithm for linear programming," *Mathematical Programming*, Vol. 44, 1989, pp. 297-335

2. A. Arbel, S.S. Oren, "Generating Search Directions in Multiobjective Linear Programming Problems Using the Analytic Hierarchy Process", *Journal of Socio-Economic Planning sciences*, Vol. 20, Number 6, pp. 369-374, 1986.

3. E.R. Barnes, "A variation on Karmarkar algorithm for solving linear programming problems," *Mathematical Programming*, Vol. 36, 1986, pp. 174-182.

4. V. Chvatal, *Linear Programming*, W.H. Freeman, 1983.

5. G.B. Dantzig, *Linear programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963.

6. P.E. Gill W. Murray and M.A. Saunders, "Interior-point methods for linear programming: A challenge to the simplex method," Technical Report SOL 88-14, Department of Operations Research, Stanford University, July 1988.

7. G.H. Golub and C.F Van Loan, *Matrix Computations*, 2nd edition, The John Hopkins University Press, Baltimore, MD. 1989.

8. N.K. Karmarkar, "A new polynomial time algorithm for linear programming," *Combinatorica*, 1984, Vol. 4, pp. 373-395.

9. N.K. Karmarkar J.C. Lagarias L. Slutsman and P. Wang, "Power series variants of Karmarkar-type algorithms," *AT&T Technical Journal*, Vol. 68, No. 3, May/June 1989, pp. 20-36.

10. M. Kojima, S. Mizuno and A. Yoshise, "A Primal-dual interior point algorithm for linear programming," *Progress in Mathematical Programming, Interior-Point and Related Methods*, N. Megido (ed.), Springer-Verlag, 1989, pp. 29-48.

11. T.L. Saaty, T.L., *Multicriteria Decision Making: The Analytic Hierarchy Process*, RSW Publications, Pittsburgh, PA., 1988.

12. T.L. Saaty and Luis G. Vargas, *The Logic of Priorities*, Kluwer-Nijhoff Publishing, 1982.

13. Steuer, R. E., *Multiple Criteria Optimization: Theory, Computation, and Application*, Wiley, 1986.

14. R.J. Vanderbei M.S. Meketon and B.A. Freedman, "A modification of Karmarkar's linear programming algorithm," *Algorithmica*, Vol. 1, 1986, pp. 395-407.

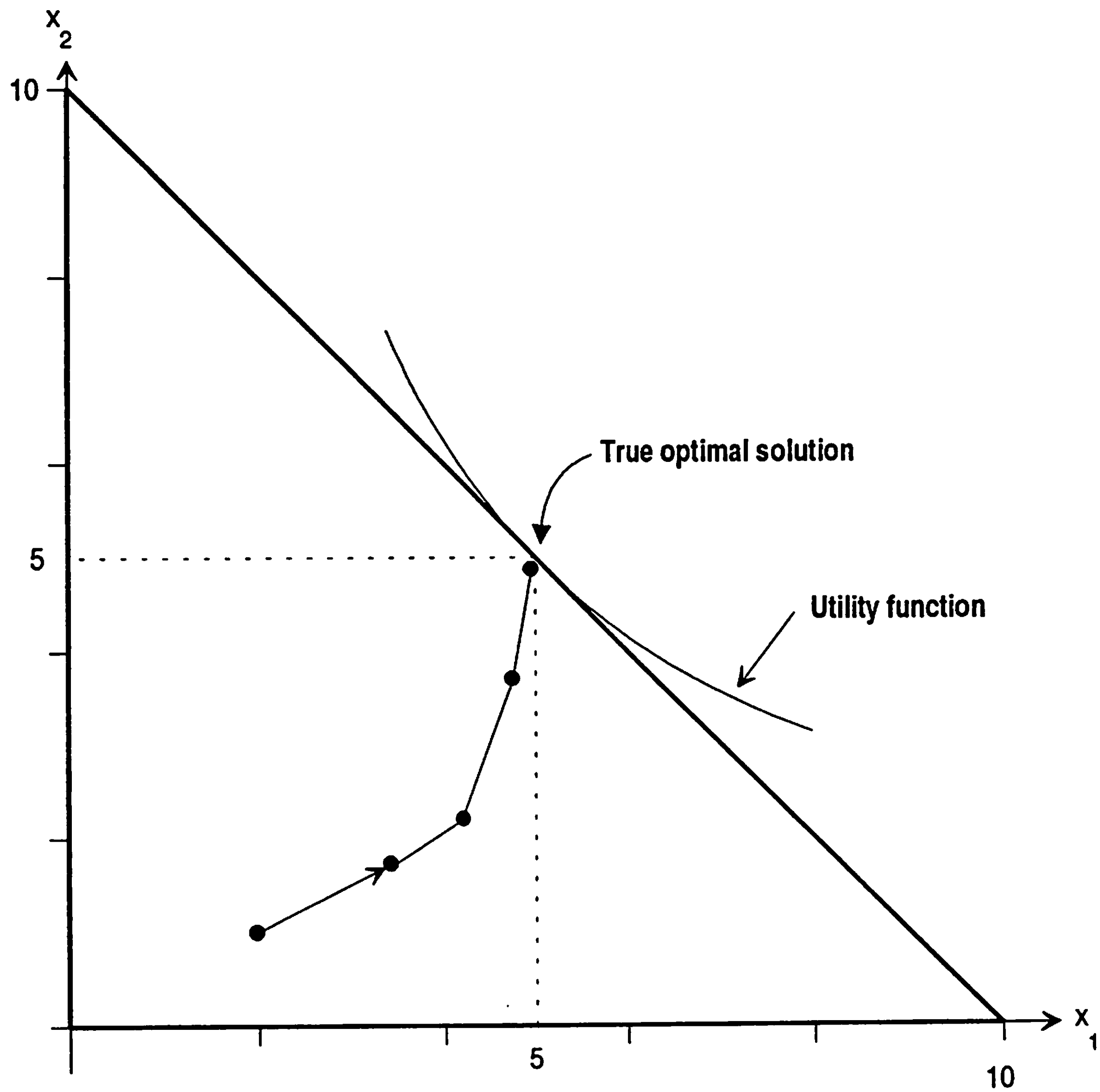

Fig. 1, The solution trajectory

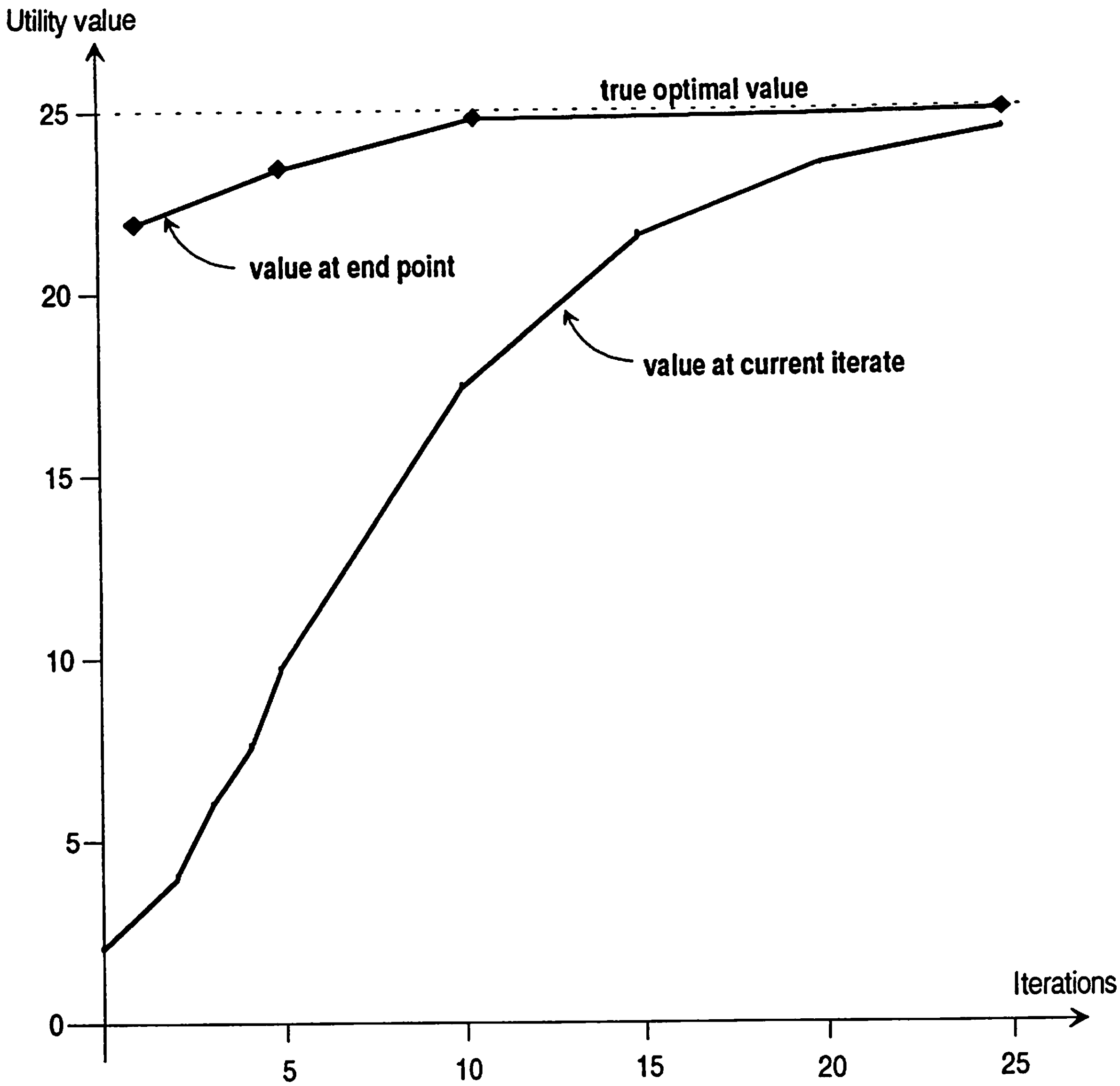

Fig. 2, The utility value along the solution trajectory