Efficient Interactive Methods for a Class of Multiattribute Shortest Path Problems

Mordechai I. Henig
Faculty of Management, Tel Aviv University, Tel Aviv, Israel

Given an acyclic network and a preference-order relation on paths, when and how can Bellman's principle of optimality be combined with interactive programming to efficiently locate an optimal path? We show that if preferences are defined via a collection of attributes, then, under common conditions, the principle of optimality is valid if and only if the preferences can be represented by a linear (value) function over the attributes. Consequently, an interactive programming method is suggested which assesses the value function while using the principle of optimality to efficiently search for an optimal path.
(*Multiple Criteria Decision Analysis; Multiple Criteria Dynamic Programming; Multiattribute Utility*)

1. Introduction

In the shortest-path problem—a cornerstone of the operations research literature—each arc connecting two nodes on a network is associated with a known length, and the objective pursued is to minimize the total length between two prescribed nodes: the source and the sink. The standard algorithm (Dreyfus 1969) for finding a shortest path in such a network amounts to solving a functional equation, based on Bellman's (1957) principle of optimality.

The importance of this problem lies in the fact that it appears in many applications (Deo and Pang 1984), in many cases as a subproblem. Moreover, the problem encompasses all finite and deterministic dynamic programming models (Denardo 1982), and its solution's algorithm is simple and efficient.

As to the input data of the problem, we can distinguish between the structure of the network, which constitutes the alternatives available to the decision maker (DM), and the length of the arcs, which constitute the objective function. As length is a known attribute for each arc, the problem is well-defined, in the sense that the DM does not have to become involved in the optimization procedure.

The trouble is that such a clear attribute does not always exist or it does not fully express the preferences of the DM. In many applications, attractiveness of paths can be captured only by a collection of attributes which, a priori, cannot be aggregated. The attributes can be cardinal—like travel cost, travel time and probability of an accident—or ordinal—like comfort and landscape. In such cases, the DM must take an active role in locating an optimal path, by revealing his/her preferences. We call this involvement "interactive programming", a procedure whereby the DM reveals the preferences concerning the paths or the attributes.

Generally speaking, however, the preferences expressed by the DM may violate the assumptions underlying the principle of optimality, in which case "Bellman's equations" do not necessarily hold; hence, locating the optimal path may require an exhaustive search over all paths.

The problem we address in this paper is: Given an acyclic network and a preference-order relation defined through a given collection of attributes, when and how can the principle of optimality be combined with interactive programming to efficiently locate an optimal path. We claim that under common conditions the principle of optimality is valid if and only if the preferences can be represented by a linear (value) function over the attributes. We also suggest an interactive programming method which assesses the value function and uses the principle of optimality to efficiently search for an optimal path.

0025-1909/94/4007/0891$01.25

Applications of the principle of optimality to general preference relations have been suggested by Brown and Strauch (1965), Mitten (1974), Sobel (1975) and Henig (1985). Mitten even describes how to handle his algorithm interactively, noting that the preferences can be expressed with respect to attributes like cost, return and risk. However, no attempt is made by the abovementioned references to verify whether the preference order underlying their algorithms implies the existence of any value function. Here, we show that if Mitten and Sobel's preferences are defined on a given set of attributes then the preference order can be presented by a linear value function over the attributes.

Fishburn (1970) and Keeney and Raiffa (1976) are but two examples of the vast literature associating a value function to an ordered set; the latter study suggests a number of methods to assess a value function over the attributes before starting optimization.

The method suggested here combines the assessment of a linear function while generating the tree of shortest paths, as is done in solving the standard shortest-path problem. The resulting algorithm has the same complexity as the standard algorithm, and interaction is kept to the required minimum.

The main thrust of the literature on interactive programming is the design of methods which converge to an "optimal" solution. Assessing a value function is usually secondary in importance (for references on this subject, see Chankong and Haimes (1983), Sawaragi et al. (1985), and Yu (1985)). The shortest-path problem with an implicit linear value function can be solved by any interactive method in linear programming, such as those suggested by Zionts and Wallenius (1976) or Steuer (1986). However, their algorithms may generate many nondominated paths before converging to an optimal one.

A comparison between algorithms which generate Pareto paths was conducted by Brumbaugh-Smith and Shier (1989). Recently, two interactive algorithms to find a shortest path have been suggested in the literature. In both cases, however, a nonlinear value function is assumed. Current et al. (1990) suggest using a method based on Handler and Zang (1980) and Henig (1986) to locate the optimal path interactively, a possible reduction in computing time being attained by using revealed tradeoffs among the attributes. Carraway et al. (1990) use heuristics to generate a subset of the nondominated paths to minimize a known nonlinear value function.

The present paper is organized as follows: In the next section, the network, the preference order and Bellman's equations to find nondominated or optimal paths are introduced. In §3 we prove that under common conditions there exists a linear value function that represents the preferences. A method to assess the function while constructing an optimal path is suggested in §4. The bicriteria case, with an example, is given in §5, followed by a summary in the final section.

2. Solving Bellman's Equations with Multiattributes

Let $G = (N, A)$ be a *network* with a finite set of *nodes* N and *arcs* $A \subset N \times N$. For the sake of simplicity, the network is assumed to be *acyclic* so that $(i, j) \in A$ only if $i > j$, the source is node $n = |N|$ and the sink is node 1. A *path* is a sequence

$$\{(i_1, i_2), (i_2, i_3), \ldots, (i_{m-1}, i_m)\}$$

of arcs where $i_k > i_{k+1}$.

Denote by $P(i)$ the set of all paths from node i to the sink. In particular, we are interested in the set $P(n)$, from which an optimal path is sought. Denote $P \equiv \cup P(i)$.

Each arc (i, j) is associated with a vector of m attributes

$$x(i, j) = (x_1(i, j), \ldots, x_m(i, j)).$$

We assume that the attributes are *additive* so that the value of a path p is the componentwise sum

$$x(p) \equiv \sum_{(i,j)\in p} (x_1(i, j), \ldots, x_m(i, j)).$$

When $m = 1$, an optimal path from node i to the sink is $p(i) = (i, p(j(i)))$, where $j(i)$ is found by recursively solving Bellman's equations

$$f(i) = x(i, j(i)) + f(j(i))$$
$$= \text{opt}\{x(i, j) + f(j): j < i\},\ i > 1,\ f(1) = 0.$$

With $m > 1$ the equations are still operational (f and x being vectors), provided there exists a well-defined op-

timization operator which represents the preference order among vectors of attributes.

Let S be a rectangle in R^m such that $[x(p): p \in P] \subset S$. We assume that the preference among paths is induced by an *asymmetric* binary relation in S. We use $\succ$ to denote the binary order relations in both P and S, since the distinction is apparent from the context. Thus for $p, q \in P(i) p \succ q$ if $x(p) \succ x(q)$.

A path $p \in P(i)$ is said to be *nondominated* if $q \not\succ p$ for all $q \in P(i)$. Brown and Strauch (1965) have extended Bellman's equations to find the set of nondominated paths. Sufficient conditions for the equations to generate the required set are:

Transitivity: for all $p, q, r \in P(i)$, if $p \succ q \succ r$ then $p \succ r$.

Strict persistence: for every $j \in N$ and $p, q \in P(j)$, $p \succ q$ implies that $(i, p) \succ (i, q)$ for every $(i, j) \in A$.

An example of an order with these properties is the *Pareto order*: $x \succ y$ if and only if $x < y$ ($x_i \leq y_i$ for all i and $x \neq y$). The nondominated paths with this order are the *Pareto paths*.

The main deficiency of any algorithm that solves the equations is its complexity, which is bounded only by the number of paths. The obvious reason for this is that the concept of nondominance is much coarser than that of optimality.

A property which allows a definition of an optimal (in the usual sense) path is *negative transitivity*: for all $p, q, r \in P(i)$, if $r \not\succ q \not\succ p$ then $r \not\succ p$. An order which is asymmetric and negatively transitive is called a *weak order*.

Denote $p \sim q$ when $q \not\succ p$ and $p \not\succ q$, and $p \succcurlyeq q$ when $p \succ q$ or $p \sim q$. When the order is weak it can be shown that $\sim$ is an equivalence relation so that if $p \succcurlyeq r \succcurlyeq q$ then $p \succcurlyeq q$. A path $p \in P(i)$ is said to be *optimal* if $p \succcurlyeq q$ for all $q \in P(i)$.

An extension (or rather an interpretation) of Bellman's equations was suggested by Mitten (1974) and Sobel (1975) for the case of a weak order. They show that every solution of Bellman's equations is an optimal path under the following condition. *Weak persistence*: for every $j \in N$ and $p, q \in P(j)$, $q \succcurlyeq p$ implies that $(i, q) \succcurlyeq (i, p)$ for every $(i, j) \in A$.

We apply the term Bellman-Mitten's algorithm (or BMA for short) to the algorithm which solves Bellman's equations to find the optimal path with multiattributes. Thus, in the multiattributes-path problem, if the order in S is weak then an optimal path can be found using the BMA if the following condition is satisfied. *Persistence (in attributes)*: for every $x, y \in S$ and $z \in R^m$, $x \succ y$ implies $x + z \succ y + z$ whenever $x + z, y + z \in S$.

Notice that $x \sim y$ implies that $x + z \sim y + z$, otherwise $x + z \succ y + z$ and by the persistence condition $x = x + z - z \succ y + z - z = y$, which is a contradiction. Thus the persistence condition here is both strict and weak.

The main deficiency of the BMA is that contrary to the standard shortest-path algorithm, Bellman's equations can be solved only by interaction with the DM, who selects an optimal path (or vector of attributes) from a set of available paths (or vectors). When the number of nodes is large, this is quite burdensome.

3. The Existence of a Linear Value Function

We show that if a value function captures the order in S then this function must be linear. More precisely, we show that if the order in S is weak, monotone, continuous and persistent then there exists a linear value function over S which expresses this order.

An order is *monotone* if for every $x, y \in S$, $x \succ y$ when $x < y$.

An order is *continuous* if for every $x, y \in S$, $x \succ y$ implies that $x + B_1 \succ y + B_2$ for some neighborhoods B_1 of x and B_2 of y.

We use Theorem 3.3 of Fishburn (1970), which states that if $\succ$ on S is weak, monotone and continuous then there exists a monotone and continuous function u on S such that $u(x) < u(y)$ if and only if $x \succ y$, $x, y \in S$. Consequently, for every $p, q \in P(i)$, $p \succ q$ if and only if $u(x(p)) < u(x(q))$.

THEOREM. *If $\succ$ is weak, monotone, continuous and persistent in S (and the attributes are additive) then there exists a linear function v on S such that $v(x) < v(y)$ if and only if $x \succ y$.*

PROOF. Without loss of generality, S is the unit cube in R^m, $u(1, \ldots, 1) = 1$ and $u(0, \ldots, 0) = 0$. By the monotonicity and continuity of u, for every $x \in S$ there exists a unique point $(r, \ldots, r)$ on the line connecting $(0, \ldots, 0)$ and $(1, \ldots, 1)$ such that $u(x) = u(r, \ldots, r)$. Accordingly, let $v(x) = r$. Clearly, v is equivalent to u, i.e., $v(x) < v(y)$ if and only if $x \succ y$.

To show that

$$v(x + y) = v(x) + v(y) \quad \text{for} \quad x, y, x + y \in S,$$

let $v(x) = r_1$ and $v(y) = r_2$. Hence,

$$x \sim (r_1, \ldots, r_1) \quad \text{and} \quad y \sim (r_2, \ldots, r_2)$$

and by the persistence condition

$$x + y \sim (r_1, \ldots, r_1) + (r_2, \ldots, r_2)$$
$$= (r_1 + r_2, \ldots, r_1 + r_2)$$

so that

$$v(x + y) = r_1 + r_2 = v(x) + v(y).$$

Notice that for every natural number k, $v(x) = kv(x/k)$, hence $v(\alpha x) = \alpha v(x)$ for every rational number α, and by the continuity condition,

$$v(\alpha x + (1 - \alpha)y) = \alpha v(x) + (1 - \alpha)v(y)$$

for every $0 \le \alpha \le 1$, and hence v is a linear function. □

The Theorem actually states that the BMA is useless if a linear value function v does not exist, since only the persistence condition assures its convergence to an optimal path. On the other hand, if such a linear function exists then the standard shortest-path algorithm can be applied after the coefficient of v has been assessed.

The results are proved here for the additive case. However, they are also true if one or more attributes are separable with respect to the product operator by taking the logarithm of the attribute. For example, consider the case when C is cost and Q is the probability of not having an accident. The persistence condition means that for every three pairs of cost and probability (C_i, Q_i), $i = 1, 2, 3$, $(C_1, Q_1) \succ (C_2, Q_2)$ implies that $(C_1 + C_3, Q_1Q_3) \succ (C_2 + C_3, Q_2Q_3)$. If a value function exists and this persistence condition is valid then there exists a real number b such that for every (C_1, Q_1) and (C_2, Q_2), $(C_1, Q_1) \succ (C_2, Q_2)$ if and only if

$$C_1 + b \log Q_1 < C_2 + b \log Q_2.$$

4. Verification and Assessment

Of all the conditions required in the Theorem, the one most pertinent to the model discussed here is the persistence condition. We can think of three approaches to verify it in an interaction with the DM.

(a) Asking the DM directly whether it is acceptable. For example, if the attributes are travel cost and travel time, an appropriate question would be "being in some node, can accumulated cost and time make a difference in the selection of the remaining path to the sink?". If the answer is negative then the persistence condition is satisfied, at least hypothetically. It can be statistically checked by the next two approaches.

(b) Presenting the DM with several pairs of $x, y \in S$ and verifying that whenever $x \succ y$ then $x + z \succ y + z$ for every $x + z, y + z \in S$.

(c) As in the previous approach, but the pairs are actual values of paths, preferably Pareto, in the network. A verification step can be introduced into the BMA by letting the DM order values $x + z$ and $y + z$, given that x and y have already been ordered.

Suppose now that a linear value function is valid. Three similar approaches can be applied to assess its coefficients.

(a) Asking the DM about the tradeoffs. In the cost-time case, this means asking for the dollar value of time.

(b) Presenting the DM with pairs of $x, y \in S$ and letting him/her order them according to the preferences.

(c) As in the previous approach, but the pairs are actual attribute values of paths, preferably Pareto, in the network.

Methods to assess a value function, according to the second approach, are common in the utility theory literature (e.g., Keeney and Raiffa 1976). Since assessment is done before starting optimization, its accuracy is determined independently of the problem and may be less than actually required or more, and thus wasteful.

Methods which follow the third approach are common in the MCDM literature (e.g., Steuer 1986, Zionts and Wallenius 1976). Many of these methods are specially tailored to optimize an implicit linear value function over a polyhedron, which is the case here. These methods, however, do not utilize the structure of the network, as the BMA does. Furthermore, the methods are not efficient, since the number of comparisons between paths is bounded only by the number of the Pareto paths, which can be extremely large. Some features of these algorithms, however, do appear in the algorithm which follows Kornbluth (1985), and which is incorporated into the BMA.

In each iteration of the BMA an "uncertainty interval" of the coefficients is given so that it is certain that the

"true" coefficients are in the interval, if the DM has not, meanwhile, changed his/her preference order. The algorithm starts out by assuming only that the function is monotone, that is, the coefficients are in the positive orthant $H = [b \in R^m: b \geq 0, \sum b_i = 1]$.

Using the recursive scheme of the BMA at each node i an optimal path $p^*(i)$ is calculated by solving

$$f(i) = \text{opt}\{x(i, j) + f(j): j < i\}.$$

$f(i)$ can be calculated by pairwise comparisons of the vectors $[x(i, j) + f(j): j < i]$, as follows.

Given two vectors x and y in R^m, if $by < bx$ for all $b \in H$ then $y \succ x$; if $bx < by$ for all $b \in H$ then $x \succ y$. Otherwise, let the DM reveal if $x \succ y$, in which case H is replaced by $H \cap [b \in R^m: b(x - y) \leq 0]$, or $y \succ x$, in which case H is replaced by $H \cap [b \in R^m: b(x - y) \geq 0]$.

The problem of whether there exists $b \in H$ such that $bx = by$ can be formulated as the linear programming problem:

$$\begin{aligned} \text{LP}(x, y, H): z &= \min\{|b(x - y)|: b \in H\} \\ &= \min\{w_1 + w_2: w_1 - w_2 - b(x - y) \\ &\qquad = 0, b \in H, w_1 \geq 0, w_2 \geq 0\}. \end{aligned}$$

Let b^* solve $\text{LP}(x, y, H)$. If $z = 0$ then $b^*x = b^*y$. If $b^*(x - y) > 0$ then $y \succ x$ and if $b^*(x - y) < 0$ then $x \succ y$.

5. The Bicriteria Case, with an Example

The algorithm is simple when $m = 2$, because the coefficients are $b = b_1$ and $b_2 \equiv 1 - b$, and the uncertainty interval of b is $[e, f]$, $0 \leq e < f \leq 1$. Initially $e = 0$ and $f = 1$. The programming problem is reduced to verifying that $bx < by$ for both $b = e$ and $b = f$, in which case $x \succ y$ or $bx > by$ for both $b = e$ and $b = f$, and hence $y \succ x$. Otherwise, according to the DM's preference, either e or f is replaced by b^*, where $b^*(x - y) = 0$.

The following procedure finds the optimal path simultaneously with the uncertainty interval for an acyclic network, after an appropriate labeling of the nodes.

Input: (c_{ij}, t_{ij}) for each arc (i, j), $j < i = 2, \ldots, n$.

Output: (c_i, t_i) the value of the optimal path, and $m(i)$ the first node from i on that path, $i = 1, 2, \ldots, n$; (e, f) the uncertainty interval.

Procedure BM:

1: $(c_1, t_1) = 0$. $i = 2$, $e = 0$, $f = 1$.

2: If $i = n + 1$ terminate. Otherwise, $k = l = 1$.
For nodes $j < i$: $C(j) = c_{ij} + c_j$, $T(j) = t_{ij} + t_j$.

3: $u_1 = eC(k) + (1 - e)T(k)$, $v_1 = fC(k) + (1 - f)T(k)$.

4: $l = l + 1$.
If $l = i$ then $(c_i, t_i) = (C(k), T(k))$, $m(i) = k$, $i = i + 1$, goto 2.
Otherwise, $u_2 = eC(l) + (1 - e)T(l)$, $v_2 = fC(l) + (1 - f)T(l)$.

5: If $u_1 \leq u_2$ and $v_1 \leq v_2$ then goto 4.
If $u_1 \geq u_2$ and $v_1 \geq v_2$ then $k = l$, $m(i) = k$, goto 3.

6: Let b solve $bC(k) + (1 - b)T(k) = bC(l) + (1 - b)T(l)$

7: *Interaction with DM*:
If $(C(k), T(k)) \succ (C(l), T(l))$ then,
if $u_1 > u_2$ and $v_1 < v_2$ then $e = b$,
if $u_1 < u_2$ and $v_1 > v_2$ then $f = b$.
If $(C(l), T(l)) \succ (C(k), T(k))$ then $k = l$,
if $u_1 < u_2$ and $v_1 < v_2$ then $f = b$,
if $u_1 < u_2$ and $v_1 > v_2$ then $e = b$.
Goto 3.

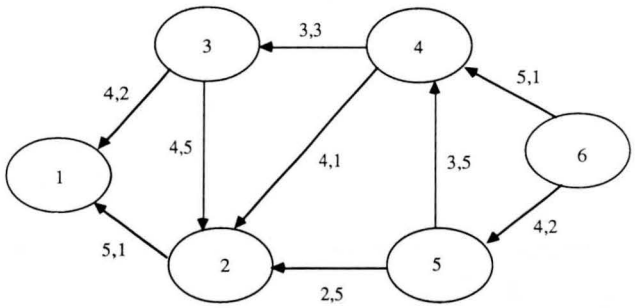
Since this procedure follows the steps of the BMA, $o(n^2)$ comparisons are performed, some of them via interaction with the DM. In any case, the uncertainty interval is not larger than it needs to be for determining the optimal path. The actual number of comparisons will be relatively small, as the polyhedron H is reduced after each comparison.

Further reduction in interaction is possible by first calculating and sorting the intersection points of the lines $[bx(i, j): j < i]$, $e \leq b \leq f$ and using a binary search to reduce the uncertainty interval, as suggested by Megiddo (1979). Then at most $o(n \log n)$ comparisons are required by the DM.

Suppose that at each iteration b^* is randomly picked in the interval (e, f) and $b \in (e, b^*)$ with probability proportional to (e, b^*). Then it is simple to show that the expected length of the new interval is $\frac{2}{3}$ of the previous one. Thus the size of the interval is approximately $(2/3)^x$ after x updates of the interval.

We now demonstrate Procedure BM for the acyclic network presented in Henig (1986) with six nodes and nine arcs (see Figure 1). Each arc is associated with two attributes. There are 3 Pareto paths from node 6 to node 1 with values (12, 6), (14, 3), (11, 8).

Figure 1 Example Network



We tried the BM procedure with preferences expressed first by $b = 0.5$, then by $b = 0.65$ and finally with $b = 0.8$.

With $b = 0.5$ we got

$(c_2, t_2) = (5, 1), m(2) = 1; (c_3, t_3) = (4, 2), m(3) = 1.$

$(c_4, t_4) = (C(2), T(2)) = (9, 2) \succ (7, 5) = (C(3), T(3)),$ $m(4) = 2, (e, f) = (0, 0.6)$

$(c_5, t_5) = (7, 6), m(5) = 2.$

$(c_6, t_6) = (14, 3), m(6) = 4.$

With $b = 0.65$ as for $b = 0.5$ except

$(c_4, t_4) = (C(3), T(3)) = (7, 5) \succ (9, 2) = (C(2), T(2)),$ $m(4) = 3, (e, f) = (0.6, 1)$

$(c_6, t_6) = (C(4), T(4)) = (12, 6) \succ (11, 8) = (C(5),$ $T(5)), m(6) = 4, (e, f) = (0.6, 0.66667).$

With $b = 0.8$ as for $b = 0.65$ except

$(c_6, t_6) = (C(5), T(5)) = (11, 8) \succ (12, 6) = (C(4),$ $T(4)), m(6) = 5, (e, f) = (0.66667, 1).$

We applied the BMA to 455 independent acyclic and complete networks, each containing up to 1,000 nodes. All the arcs (i, j) with $i > j$ exist, each having two attributes $c_{ij} = x(i - j)$ and $t_{ij} = y(i - j)$ where x and y are independent uniform numbers on $[0, 1]$. In 339 networks a linear value function with $b = 0.5$ was assumed to simulate the DM's preferences. In the remaining 116 networks other values of b: 0.1, 0.2, 0.3 and 0.4 were assumed.

The number of interactions necessary to find an optimal path was related to the number of nodes in the network. A log function with no intercept was found to be a good (least square) predictor:

number of interactions

$$= 6.7 * \log_{10}(\text{nodes}) \approx 1 + 3.4 * \log_{10}(\text{arcs}),$$

with standard deviation $= 1.82 * \log_{10}(\text{nodes})$.

There were no significant differences between the various values of b. Thus, on average, when the number of nodes in the network is doubled, only 2 additional comparisons have to be accomplished by the DM. We also found that the length of the uncertainty interval (which is 1 at the beginning) is $3.71 * (\text{nodes})^{-1.63}$ at the end of the procedure.

6. Conclusions

The main issue in dynamic programming is usually not one of formulation but of application. As Mitten and Sobel have shown, an ordinal functional equation can be formulated under common conditions. As argued here, such a formulation is not helpful in real cases. However, if the common conditions can be projected into R^m, by introducing m additive attributes, then preferences can be expressed by a linear value function. The result is an algorithm which is essentially the standard efficient shortest-path algorithm. The number of interactions with the DM is kept to the minimum. As the simulation with two attributes shows, the number of interactions increases as a log function of the network size, suggesting a relatively small number of interactions. Further investigation is required to relate the number of interactions to the network size for networks with more than two attributes.[1]

[1] The work originated as an Ms.c. thesis (in Hebrew) by Tamar Solomon, The Faculty of Management, Tel Aviv University, 1990.

References

Bellman, R., *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

Brown, T. A. and R. E. Strauch, "Dynamic Programming in Multiplicative Lattices," *J. Math. Anal. Appl.*, 12 (1965), 364–370.

Brumbaugh-Smith, J. and D. Shier, "An Empirical Investigation of Some Bicriterion Shortest Path Algorithms," *European J. Operational Res.*, 43 (1989), 216–224.

Carraway, R., L. Morin and H. Moskowitz, "Generalized Dynamic Programming for Multicriteria Optimization," *European J. Operational Res.*, 44 (1990), 95–104.

Chankong, V. and Y. Y. Haimes, *Multiobjective Decision Making: Theory and Methodology*, North Holland, New York, 1983.

Current, J. R., S. R. Charles and J. L. Cohon, "An Interactive Approach to Identify the Best Compromise Solution for Two Objective Shortest Path Problems," *Computers Oper. Res.*, 17 (1990), 187–198.

Denardo, E. V., *Dynamic Programming, Models and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1982.

Deo, N. and C. Pang, "Shortest-Path Algorithms: Taxonomy and Annotation," *Networks*, 14 (1984), 275–323.

Dreyfus, S. E., "An Appraisal of Some Shortest-Path Algorithms," *Operations Res.*, 17 (1969), 395–412.

Fishburn, P. C., *Utility Theory for Decision Making*, Wiley, New York, 1970.

Handler, G. and I. Zang, "A Dual Algorithm for the Constrained Shortest Path Problem," *Networks*, 10 (1980), 293–310.

Henig, M. I., "The Principle of Optimality in Dynamic Programming with Returns in Partially Ordered Sets," *Mathematics of Operations Res.*, 10 (1985), 462–470.

——, "The Shortest Path With Two Objective Functions," *European J. Operational Res.*, 25 (1986), 281–291.

Keeney, L. R. and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, John Wiley & Sons, New York, 1976.

Kornbluth, J. S. H., "Sequential Multi-Criterion Decision Making," *Omega*, 13 (1985), 569–574.

Megiddo, N., "Combinatorial Optimization with Rational Objective Functions," *Mathematics of Operations Res.*, 4 (1979), 414–424.

Mitten, L. G., "Preference Order Dynamic Programming," *Management Sci.*, 21 (1974), 43–46.

Sawaragi, Y., H. Nakayama and T. Tanino, *Theory of Multiobjective Optimization*, Academic Press, San Diego, CA, 1985.

Sobel, M. J., "Ordinal Dynamic Programming," *Management Sci.*, 21 (1975), 967–975.

Steuer, R., *Multiple Criteria Optimization: Theory, Computation, and Application*, John Wiley & Sons, New York, 1986.

Yu, P. L., *Multiple Criteria Decision Making: Concepts, Techniques and Extensions*, Plenum, New York, 1985.

Zionts, S. and J. Wallenius, "An Interactive Programming Method for Solving the Multiple Criteria Problem," *Management Sci.*, 22 (1976), 652–663.

Accepted by Thomas M. Liebling; received January 31, 1991. *This paper has been with the author* 4 *months for* 3 *revisions.*