RAIRO. RECHERCHE OPÉRATIONNELLE

TEODROS GETACHEW
MICHAEL KOSTREVA
LAURA LANCASTER

A generalization of dynamic programming for Pareto optimization in dynamic networks

RAIRO. Recherche opérationnelle, tome 34, n° 1 (2000), p. 27-47
<http://www.numdam.org/item?id=RO_2000__34_1_27_0>

RAIRO Operations Research
RAIRO Oper. Res. **34** (2000) 27-47

A GENERALIZATION OF DYNAMIC PROGRAMMING FOR PARETO OPTIMIZATION IN DYNAMIC NETWORKS*

by Teodros GETACHEW ([1]), Michael KOSTREVA ([1]) and Laura LANCASTER ([1])

Communicated by Franco GIANNESSI

Abstract. – *The Algorithm in this paper is designed to find the shortest path in a network given time-dependent cost functions. It has the following features: it is recursive; it takes place bath in a backward dynamic programming phase and in a forward evaluation phase; it does not need a time-grid such as in Cook and Halsey and Kostreva and Wiecek's "Algorithm One"; it requires only boundedness (above and below) of the cost functions; it reduces to backward multi-objective dynamic programming if there are constant costs. This algorithm has been successfully applied to multi-stage decision problems where the costs are a function of the time when the decision is made. There are examples of further applications to tactical delay in production scheduling and to production control.*

Keywords: Pareto optimization, dynamic network, shortest path, dynamic programming, time-dependent cost function.

1. INTRODUCTION

Since its formulation by Bellman [1], dynamic programming has proven to be a useful tool in the solution of multi-stage decision problems. However, despite its popularity and usefulness in many applications, dynamic programming, as originally introduced, has many limitations. Over the last few decades, there have been numerous efforts to expand the use of dynamic programming by generalizing it or changing it slightly to fit certain applications.

Brown and Strauch [2] generalized dynamic programming by allowing the range of decision functions to be a regular multiplicative lattice. In

(*) Received July 1996.

([1]) Clemson University, Department of Mathematical Sciences, Clemson, South Carolina 29634-1907, U.S.A.

Recherche opérationnelle/Operations Research, 0399-0559/00/01/$ 7.00

similar work, Henig [13] investigated the Principle of Optimality when the returns are in a partially ordered set. Verdu and Poor [18] proposed an abstract Dynamic Programming model that includes, but is not restricted to, optimization problems. Carraway *et al.* [4] introduced a way to use generalized DP with a multicriteria preference function. Basically, they used a weaker Principle of Optimality than Bellman's original version.

Daellenbach and DeKluyver [7] were the first to work on Dynamic Programming with explicit multiple objectives. They introduced a computational method to find Pareto Optimal paths through a network, with the restriction that the distance from a given node to the destination node is unique. Their method is a straightforward extension of the Principle of Optimality to the multiple objective context. The method they proposed is emulated in many of the subsequent papers on multiple objective Dynamic Programming. Shortly thereafter, Corley and Moon [6] presented almost the same algorithm as Daellenbach and DeKluyver with the only difference being that the vector-minimization takes place over paths of k or fewer links instead of paths with exactly k links.

Cooke and Halsey [5] proposed one of the first algorithms using Dynamic Programming in a time-dependent context. Their application was a routing problem with time-dependent transition times between states. Other papers that do not use dynamic programming to solve time-dependent shortest path problems, but are of interest here do exist. In a general survey of shortest-path algorithms, Dreyfus [9] briefly discussed the problem of finding shortest paths in networks with time-dependent arc lengths. He proposed a modification of Dijkstra's [8] famous shortest path algorithm. Philpott [17] attacked a similar problem using a continuous-time linear programming formulation. Halpern [12] proposed an algorithm to determine the shortest route in a network with edge transit times that are time varying and nodes that allow limited waiting. In work related to Halpern's paper, Orda and Rom [16] considered the same problem with several different waiting ("parking") models and discussed the computational complexity of their proposed algorithms.

Both Halpern's and Dreyfus' algorithms for the solution of the shortest path problem with time-dependent cost functions are proven to fail by a counterexample in Getachew [10]. They fail because they are "memoryless". In other words, all subpaths are permanently discarded when they are not part of an optimal subpath, even though the non-optimal subpath could become part of the optimal path from the origin to the destination. This occurs because it is possible that waiting longer to reach a certain node may be

less costly than reaching it early. Thus, what may seem like a costly subpath could end up being less costly over all. Cooke and Halsey's algorithm avoids this problem because it does not discard these subpaths. Unfortunately, this also means that a great deal of memory is required, especially when their algorithm is extended to the multiple objective version.

Kostreva and Wiecek introduced two algorithms that involve both multi-objective dynamic programming and time-dependent cost functions. Their "Algorithm One" extended some of the work by Cooke and Halsey on time-dependent routing to the multi-criteria case. This is a method using backward dynamic programming and a time-grid. Under some weak assumptions, this algorithm finds all Pareto paths from every node in the network to the destination node. Their "Algorithm Two" generalized earlier work by Kaufman and Smith [14] on finding minimum travel time paths in networks with time-varying transit times. This method uses forward dynamic programming to find all Pareto paths from the origin node to every other node in the network. This algorithm does not use a time-grid, but requires that all cost functions be monotonically non-decreasing and allows no passing.

The algorithm in this paper is designed to find the shortest path in a network given time-dependent cost functions. It has the following features:

1. It is recursive.
2. It takes place both in a backward dynamic programming phase and in a forward evaluation phase.
3. It does not need a time-grid such as in Cooke and Halsey and Kostreva and Wiecek's "Algorithm One".
4. It requires only boundedness (above and below) of the cost functions, as opposed to Kostreva and Wiecek's "Algorithm Two" which requires monotonic non-decreasing cost functions.
5. It reduces to backward multi-objective dynamic programming if there are constant costs.

This algorithm has been successfully applied to multi-stage decision problems where the costs are a function of the time when the decision is made. For example, in an occupant's egress from a burning building, the decision of which path to take is dependent on the fire, smoke, toxins, etc. which are functions clearly dependent on time. In Getachew [10] there are examples of further applications to tactical delay in production scheduling and to production control.

2. ALGORITHM DEVELOPMENT

Notation, definitions, and terminology

General network

First, we consider a general network containing a set of nodes, $N = \{(1, 2, \ldots, n)\}$, and a set of links, $L = \{(i_0, i_1), (i_2, i_3), (i_4, i_5), \ldots\} \subset N \times N$, which indicates connections between nodes. An element (i, j) of L is referred to as a directed link from node i to node j. A directed network G is the ordered pair (N, L).

Paths

A path from node i to j is the set of links $\Pi = \{(i, i_1), (i_1, i_2), (i_2, i_3), \ldots, (i_{k-2}, i_{k-1}), (i_{k-1}, j)\}$, where the first node of each link is the same as the terminal node of the preceding link and each node in the path is unique. The path Π is said to have cardinality k. In this paper we shall only be interested in paths that terminate in a pre-selected node. Let $\{d\} \subset N$, the destination node, be given. Then $P_i(G)$ shall denote the set of paths Π with initial node i and final node d. The set of all paths emanating from nodes that are distinct from the destination node is denoted by $P(G)$. Note that

$$P(G) = \bigcup_{i \in N \setminus \{d\}} P_i(G).$$

Path cardinality

In the first phase of the algorithm, it is important to single out paths satisfying specific cardinality criteria.

$P_i^{(k)}(G)$ = the set of paths in $P_i(G)$ of cardinality at most k.

$P_i^k(G)$ = the set of paths in $P_i(G)$ of cardinality exactly k.

$P^{(k)}(G)$ = the set of all paths in $P(G)$ of cardinality at most k where

$$P^{(k)}(G) = \bigcup_{i \in N \setminus \{d\}} P_i^{(k)}(G).$$

$P^k(G)$ = the set of all paths in $P(G)$ of cardinality exactly k where

$$P^k(G) = \bigcup_{i \in N \setminus \{d\}} P_i^k(G).$$

Subpaths

The concept of a subpath is fundamental for the first phase of this algorithm. Let Π' be a subpath of $\Pi \in P(G)$ if and only if $\Pi' \subset \Pi$ and $\Pi' \in P(G)$. That is, a subpath, in addition to being a subset of a path, must itself terminate at node d.

Cost functions

This algorithm requires only that the cost functions be bounded. The cost functions must be bounded above to satisfy the global requirement of finite costing, and they must be bounded below as a requirement of the first phase of the algorithm. Moreover, the costs of subpaths do not have to be separable from the costs of paths that contain them.

Let F be a set of link- transition cost functions with domain $\mathbb{R}_+ \cup \{0\}$, called "time", and range $\mathbb{R}_+$, bounded above and below, the cost. These link-transition cost functions are given by the range of the function $C : L \to F$, where $C((i,j)) = c_{ij}(t) \in F$, $t \in \mathbb{R}_+ \cup \{0\}$. That is, each link can have a different time-dependent cost function.

Multiobjective cost vectors

Since the algorithm is developed for multiple objectives, we need to consider vector cost functions. Basically, these are vectors of cost functions as defined above. The p-dimensional vector cost function is given as the range of the function

$$\vec{c} : L \to F^p \text{ where } \vec{c}(i,j)(t) = (c_{ij}^1(t), \ldots, c_{ij}^p(t)),$$

where $c_{ij}^q(t) \in F$, $q = 1, \ldots, p$.

The cost of traversing a path in the network is computed by adding up the cost vectors on each link of the path. These costs, in turn, depend on time. We assume that these costs are evaluated at the time of arrival at the first node of the link (frozen link model). Thus, each link must have a "cost" function that gives the time to traverse the link. Let $c_{ij}^T(t) \in F$ be the travel time function for the link (i,j) given the arrival to node i at

time t. We do not assume that this travel time function is a part of the cost vector, although it could be.

Arrival function

A function that gives the time of arrival at each node along a path clearly depends on the path being traversed and the node of interest. Let $\Pi = \{(i, j_1), (j_1, j_2), \ldots, (j_{k-2}, j_{k-1}), (j_{k-1}, d)\}$ be a path in $P_i^k(G)$. The arrival function $A : P(G) \times N \to \mathbb{R}_+ \cup \{0\}$ is defined recursively on initial link-nodes as follows:

$$A(\Pi, i) = 0.$$

Suppose $A(\Pi, j_r)$ has been defined for $r \leq s$ and let (j_s, j_{s+1}) be an element of Π.

$$A(\Pi, j_{s+1}) = A(\Pi, j_s) + c^T_{j_s j_{s+1}}(A(\Pi, j_s)).$$

Path costing functions

As will be shown shortly, the algorithm has two main phases. The backward dynamic programming phase and the forward integration phase evaluate costs on the paths in different ways. In the former, the infima of the link transition cost functions are used, while in the latter the actual costs to traverse the paths are found.

First we define the path costing function for the path Π in the backward dynamic programming phase, Γ_0. Let $c_0^q(i, j) = \inf_{t \in [\alpha, \beta]} c_{ij}^q(t)$ where

$$\alpha = \min_{\Pi \in P(G)} \{A(\Pi, i)\} \qquad \text{and} \qquad \beta = \max_{\Pi \in P(G)} \{A(\Pi, j)\}.$$

This is the infimum of the cost function for every link given the time ranges possible for all paths. Now, let $\Pi \in P_i(G)$ where $\Pi = \{(i, j_1), (j_1, j_2), \ldots, (j_{k-2}, j_{k-1}), (j_{k-1}, d)\}$.

Then $\Gamma_0 : P(G) \to (\mathbb{R}_+)^P$ is defined by

$$\Gamma_0(\Pi) = \vec{c}_0(i, j_1) + \vec{c}_0(j_{k-1}, d) + \sum_{s=1}^{k-2} \vec{c}_0(j_s, j_{s+1}),$$

where $\vec{c}_0(i, j) = \big(c_0^1(i, j), \ldots, c_0^p(i, j)\big)$. So, Γ_0 evaluates the cost of a path given that every link is at its infimum as defined by the above c_0 function.

Next we define the path costing function Γ for the path Π in the forward integration phase. Let $\Gamma : P(G) \to (\mathbb{R}_+)^P$ be defined by

$$\Gamma(\Pi) = \vec{c}(i, j_1)(0) + \vec{c}(j_{k-1}, d)(A(\Pi, j_{k-1})) + \sum_{s=1}^{k-2} \vec{c}(j_s, j_{s+1})(A(\Pi, j_s)).$$

Pareto optimality

The algorithm of this paper finds the set of Pareto optimal solutions to the multiple objective shortest path problem. The concept of Pareto optimality for a multiple objective function is now defined:

Let $\mathrm{Eff}_i^{(k)}(P_0)$ be the set of paths of k or less links and starting at node i that are non-dominated according to the Γ_0 costing function. Let $\mathrm{Eff}_i(P_0)$ be the set of all paths starting at node i that are non-dominated according to the Γ_0 costing function. Let $\mathrm{Eff}_i^{(k)}(P)$ be the set of paths of k or less links and starting at node i that are non-dominated according to the Γ costing function. Let $\mathrm{Eff}_i(P)$ be the set of all paths starting at node i that are non-dominated according to the Γ costing function. Let $\Pi \in P^{(k)}(G)$. Then $\Pi \in \mathrm{Eff}_i^{(k)}(P_0)$ if and only if the set $\{\Pi' : \Pi' \in P^{(k)}(G), \Gamma_0(\Pi') \leq \Gamma_0(\Pi), \Gamma_0(\Pi') \neq \Gamma_0(\Gamma)\}$ is empty. Similarly, $\Pi \in \mathrm{Eff}_i^{(k)}(P)$ if and only if the set $\{\Pi' : \Pi' \in P^{(k)}(G), \Gamma(\Pi') \leq \Gamma(\Pi), \Gamma(\Pi') \neq \Gamma(\Pi)\}$ is empty. Note that $\mathrm{Eff}_i(P_0) = \mathrm{Eff}_i^{(n-1)}(P_0)$ and $\mathrm{Eff}_i(P) = \mathrm{Eff}_i^{(n-1)}(P)$.

3. THE ALGORITHM

In this section, we develop in detail a new algorithm for solving multi-objective optimization problems in networks. It is more general than those described in the introductory section, and for the case of all cost functions taking constant values, it reduces to multiple objective backward dynamic programming. An example of its application follows in Section 4.

Iteration I_0

Given a network, G, with each link having an associated transition cost vector as defined above, the algorithm begins by partitioning the set of paths. One set, the *forbidden set* S_0, is the empty set and the other set, the *working set*, is the entire set of paths with all links costed with Γ_0. Note that the Γ_0 costing yields a unique network, as each link transition cost function has a

unique infimum. What follows then is an application of backward multiple objective dynamic programming to this network. For all $i \in G$, $i \neq d$ do:

$$F_{0i}^{(1)} = VMIN\{\Gamma_0(\Pi) : \Gamma_0(\Pi) = \vec{c}_0(i,d),\ where\ \Pi \in P_i^{(1)}(G)\};$$
$$F_{0i}^{(2)} = VMIN\{\Gamma_0(\Pi) : \Gamma_0(\Pi) = \vec{c}_0(i,j) + f_{0i}^{(1)}, f_{0j}^{(1)} \in F_{0j}^{(1)}$$
$$where\ \Pi \in P_i^{(2)}(G),\ \forall\, j \in G\};$$
$$\ldots$$
$$F_{0i}^{(r)} = VMIN\{\Gamma_0(\Pi) : \Gamma_0(\Pi) = \vec{c}_0(i,j) + f_{0j}^{(r-1)}, f_{0j}^{(r-1)} \in F_{0j}^{(r-1)}$$
$$where\ \Pi \in P_i^{(r)}(G),\ \forall\, j \in G\};$$
$$\ldots$$
$$F_{0i}^{(n-1)} = VMIN\{\Gamma_0(\Pi) : \Gamma_0(\Pi) = \vec{c}_0(i,j) + f_{0j}^{(n-2)}, f_{0j}^{(n-2)} \in F_{0j}^{(n-2)}$$
$$where\ \Pi \in P_i^{(n-1)}(G),\ \forall\, j \in G\}.$$

The backward dynamic programming stage terminates with the output set $F_{0i}^{(n-1)}$. Let $F_{0i} \equiv F_{0i}^{(n-1)}$, and $P(F_{0i}) = \{\Pi : \Pi \in P(G)$ and $\Gamma_0(\Pi) \in F_{0i}\}$.

In the forward integration phase, the true cost of each path whose cost is in the set F_{0i} is now found by direct evaluation *via* the costing function Γ. Let $V_{0i} = \{\Pi : \Pi \in P(F_{0i}),\ \Gamma_0(\Pi) \leq \Gamma(\Pi)$ and $\Gamma_0(\Pi) \neq \Gamma(\Pi)\}$ be the set of paths starting at node i whose Γ_0 is *not equal* to the actual cost of traversing the path.

If V_{0i} is empty, then STOP, with $P(F_{0i}) = \mathrm{Eff}_i^{(n-1)}(P)$; otherwise let $u_{0i} = \varnothing$, let $u_{1i} = V_{0i} \cup u_{0i}$, and let $u_1 = \bigcup_{i \neq d} u_{1i}$. The forbidden set for the iteration I_1 then becomes, $S_1 = \{\Pi : \Pi \in P(G)$ such that Π is a **subpath** of Π', $\Pi' \in u_1\}$.

Iteration I_r

In general $S_r = \{\Pi : \Pi \in P(G)$ such that Π is a **subpath** of Π', $\Pi' \in u_r\}$. This forbidden set defines the partition for iteration I_r to be the paths in S_r and its complement. The path costing function for the r^{th} iteration is now given as:

$$\Gamma_r : P(G) \to (\mathbb{R}_+)^k \text{ where } \Gamma_r(\Pi) = \begin{cases} \Gamma_0(\Pi),\ \Pi \notin S_r, \\ \Gamma(\Pi) \text{ otherwise.} \end{cases}$$

Let $S_{ri} \equiv P_i(G) \cap S_r$, and $S_{ri}^k \equiv P_i^k(G) \cap S_r$.

For all $i \in G$, $i \neq d$ the backward dynamic programming phase is applied to the set of paths that are in the complement of the set S_{ri}.

$$F_{ri}^{(1)} = \text{VMIN}\{\Gamma_r(\Pi) : \Gamma_r(\Pi) = \vec{c}_0(i,d), \text{ where } \Pi \in P_i^{(1)}(G) \text{ and } \Pi \notin S_{ri}\}.$$

Paths of length at least two in the complement of S_{ri} can take two forms: those with all links but the latest resulting from the previous vector-minimization, F_{rj}, and those with all links except the latest being in a set, S_{rj}. Accordingly, vector-minimization over sets of paths of cardinality greater than one must involve not only paths from the preceding vector-minimization step, but also paths from the appropriate forbidden sets.

So, for $k = 2$ to $n-1$:

$$\begin{aligned} F_{ri}^{(k)} = \text{VMIN}\{&\{\Gamma_r(\Pi) : \Gamma_r(\Pi) = \vec{c}_0(i,j) + f_{rj}^{(k-1)}, f_{rj}^{(k-1)} \in F_{rj}^{(k-1)} \\ &\text{where } \Pi \in P_i^{(k)}(G),\ \Pi \notin S_{ri},\ \forall\, j \in G\} \\ &\cup \{\Gamma_r(\Pi) : \Gamma_r(\Pi) = \vec{c}_0(i,j) + \Gamma_0(\Pi'), f_{rj}^{(k-1)} \in F_{rj}^{(k-1)} \\ &\text{where } \Pi' \in S_{rj}^{k-1},\ \Pi \in P_i^{(k)}(G),\ \Pi \notin S_{ri},\ \forall\, j \in G\}\}. \end{aligned}$$

Let $F_{ri} = \text{VMIN}\{F_{ri}^{(n-1)} \cup \{s_{ri}\}\}$ where $\{s_{ri}\} \equiv \{\Gamma_r(\Pi) : \Pi \in S_{ri}\}$.

Let $P(F_{ri}) \equiv \{\Pi : \Pi \in P^i(G) \text{ and } \Gamma_r(\Pi) \in F_{ri}\}$ and let the set V_{ri} be defined by

$$V_{ri} = \{\Pi : \Pi \in P(F_{ri}),\ \Gamma_r(\Pi) \neq \Gamma(\Pi)\}.$$

If V_{ri} is empty, STOP.

$$P(F_{ri}) = \text{eff}_i(P).$$

Else let

$$u_{(r+1)i} \equiv V_{ri} \cup u_{ri} \qquad \text{and} \qquad u_{r+1} \equiv \bigcup_{i \neq d} u_{(r+1)i}.$$

The algorithm above, with its alternating forward and backward phases, reduces to multiple objective backward dynamic programming in the case of constant costs. An illustrative example is presented in the next section. This example features discontinuous objective functions, but also note that it is of the class solvable by [15]. Thus, the power of the method is demonstrated.

More general problems, beyond those covered by [15], are also solvable by the new algorithm. Some of these are presented in [10], while some will appear in other work in preparation. Section 5 contains the theoretical justification of the algorithm and some other results of a more general nature.

4. AN EXAMPLE

This network has a cost vector with two cost functions. Two of the links have time-dependent cost vectors, links (3,4) and (5,6). The destination node is node 6. The problem is to find all the efficient paths from all start nodes 1, 2, 3, 4, and 5 to the destination node, node 6.

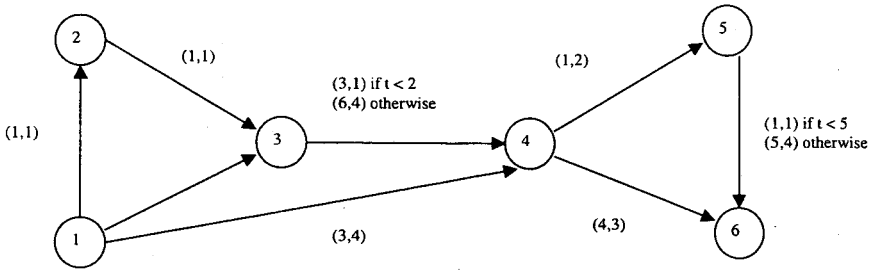

Figure 1. – A network with time-dependent link-transition cost functions.

Iteration 0

The backward dynamic programming phase finds all non-dominated paths from every node to the destination node given the costing function Γ_0.

$F_{01}^{(K)}$	Max path-length	Start node	Path	Cost
$F_{01}^{(1)}$	1	1	—	—
$F_{02}^{(1)}$	1	2	—	—
$F_{03}^{(1)}$	1	3	—	—
$F_{04}^{(1)}$	1	4	4 – 6	(4,3)
$F_{05}^{(1)}$	1	5	5 – 6	(1,1)
$F_{01}^{(2)}$	2	1	1 – 4 – 6	(3,4) + (4,3) = (7,7)
$F_{02}^{(2)}$	2	2	—	—

$F_{03}^{(2)}$	2	3	3 – 4 – 6	(3,1) + (4,3) = (7,4)
$F_{04}^{(2)}$	2	4	4 – 6 4 – 5 – 6	(4,3) (1,2) + (1,1) = (2,3)
$F_{05}^{(2)}$	2	5	5 – 6	(1,1)
$F_{01}^{(3)}$	3	1	1 – 4 – 6 1 – 3 – 4 – 6	(7,7) (1,4) + (7,4) = (8,8)
$F_{02}^{(3)}$	3	2	2 – 3 – 4 – 6	(1,1) + (7,4) = (8,5)
$F_{03}^{(3)}$	3	3	3 – 4 – 6 3 – 4 – 5 – 6	(7,4) (3,1) + (2,3) = (5,4)
$F_{04}^{(3)}$	3	4	4 – 5 – 6	(2,3)
$F_{05}^{(3)}$	3	5	5 – 6	(1,1)
$F_{01}^{(4)}$	4	1	1 – 4 – 6 1 – 3 – 4 – 5 – 6 1 – 2 – 3 – 4 – 6	(7,7) (1,4) + (5,4) = (6,8) (1,1) + (8,5) = (9,6)
$F_{02}^{(4)}$	4	2	2 – 3 – 4 – 6 2 – 3 – 4 – 5 – 6	(8,5) (1,1) + (5,4) = (6,5)
$F_{03}^{(4)}$	4	3	3 – 4 – 5 – 6	(5,4)
$F_{04}^{(4)}$	4	4	4 – 5 – 6	(2,3)
$F_{05}^{(4)}$	4	5	5 – 6	(1,1)
$F_{01}^{(5)}$	5	1	1 – 4 – 6 1 – 3 – 4 – 5 – 6 1 – 2 – 3 – 4 – 6 1 – 2 – 3 – 4 – 5 – 6	(7,7) (6,8) (9,6) (1,1) + (6,5) = (7,6)
$F_{02}^{(5)}$	5	2	2 – 3 – 4 – 5 – 6	(6,5)
$F_{03}^{(5)}$	5	3	3 – 4 – 5 – 6	(5,4)
$F_{04}^{(5)}$	5	4	4 – 5 – 6	(2,3)
$F_{05}^{(5)}$	5	5	5 – 6	(1,1)

This phase terminates with the set $F_{0i} = F_{0i}^{(5)}$.

$$P(F_{01}) = \{1-3-4-5-6,\ 1-2-3-4-5-6\}$$
$$P(F_{02}) = \{2-3-4-5-6\}$$
$$P(F_{03}) = \{3-4-5-6\}$$
$$P(F_{04}) = \{4-5-6\}$$
$$P(F_{05}) = \{5-6\}.$$

The true cost of each path in $P(F_{0i})$ is now found by direct evaluation using the Γ costing function.

Efficient path	Infimum cost	True cost
1 – 2 – 3 – 4 – 5 – 6	(6,8)	(10,11)
1 – 2 – 3 – 4 – 5 – 6	(7,6)	(14,12)
2 – 3 – 4 – 5 – 6	(6,5)	(10,8)
3 – 4 – 5 – 6	(5,4)	(5,4)
4 – 5 – 6	(2,3)	(2,3)
5 – 6	(1,1)	(1,1)

So, V_{0i} becomes:

$$V_{01} = \{1-3-4-5-6,\ 1-2-3-4-5-6\}$$
$$V_{02} = \{2-3-4-5-6\}$$
$$V_{03} = \varnothing \Rightarrow P(F_{03}) = \mathrm{Eff}_3(P)$$
$$V_{04} = \varnothing \Rightarrow P(F_{04}) = \mathrm{Eff}_4(P)$$
$$V_{05} = \varnothing \Rightarrow P(F_{04}) = \mathrm{Eff}_5(P).$$

Iteration 1

$$u_1 = \{1-3-4-5-6,\ 1-2-3-4-5-6,\ 2-3-4-5-6\}$$

So, the avoidance set for this iteration then becomes:

$$S_1 = \{1-2-3-4-5-6,\ 2-3-4-5-6,\ 3-4-5-6,\ 4-5-6,\ 5-6,\ 1-3-4-5-6\}$$
$$S_{11} = \{1-2-3-4-5-6,\ 1-3-4-5-6\}$$
$$S_{12} = \{2-3-4-5-6\}$$
$$S_{13} = \{3-4-5-6\}$$
$$S_{14} = \{4-5-6\}$$
$$S_{15} = \{5-6\}.$$

$F_{11}^{(K)}$	Max path-length	Start node	Path	Cost
$F_{11}^{(1)}$	1	1	—	—
$F_{12}^{(1)}$	1	2	—	—

$F_{13}^{(1)}$	1	3	—	—
$F_{14}^{(1)}$	1	4	4 – 6	(4,3)
$F_{11}^{(2)}$	2	1	1 – 4 – 6	(3,4) + (4,3) = (7,7)
$F_{12}^{(2)}$	2	2	—	—
$F_{13}^{(2)}$	2	3	3 – 4 – 6	(3,1) + (4,3) = (7,4)
$F_{14}^{(2)}$	2	4	4 – 6	(4,3)
$F_{11}^{(3)}$	3	1	1 – 4 – 6 1 – 3 – 4 – 6	(7,7) (1,4) + (7,4) = (8,8)
$F_{12}^{(3)}$	3	2	2 – 3 – 4 – 6	(1,1) + (7,4) = (8,5)
$F_{13}^{(3)}$	3	3	3 – 4 – 6	(7,4)
$F_{14}^{(3)}$	3	4	4 – 6	(4,3)
$F_{11}^{(4)}$	4	1	1 – 4 – 6 1 – 2 – 3 – 4 – 6	(7,7) (1,1) + (8,5) = (9,6)
$F_{12}^{(4)}$	4	2	2 – 3 – 4 – 6	(8,5)
$F_{13}^{(4)}$	4	3	3 – 4 – 6	(7,4)
$F_{14}^{(4)}$	4	4	4 – 6	(4,3)
$F_{11}^{(5)}$	5	1	1 – 4 – 6 1 – 2 – 3 – 4 – 6	(7,7) (9,6)
$F_{12}^{(5)}$	5	2	2 – 3 – 4 – 6	(8,5)
$F_{13}^{(5)}$	5	3	3 – 4 – 6	(7,4)
$F_{14}^{(5)}$	5	4	4 – 6	(4,3)

This phase terminates with the set $F_{1i} = \mathrm{VMIN}\{F_{1i}^{(5)} \cup \{s_{1i}\}\}$.

$$F_{11} = \mathrm{VMIN}\{(7,7),(9,6),(10,11),(14,12)\} = \{(7,7),(9,6)\}$$
$$F_{12} = \mathrm{VMIN}\{(8,5),(10,8)\} = \{(8,5)\}$$
$$F_{13} = \mathrm{VMIN}\{(7,4),(5,4)\} = \{(5,4)\}$$
$$F_{14} = \mathrm{VMIN}\{(4,3),(2,3)\} = \{(2,3)\}$$
$$F_{12} = \mathrm{VMIN}\{(1,1)\} = \{(1,1)\}$$
$$P(F_{11}) = \{1-4-6,\ 1-2-3-4-6\}$$
$$P(F_{12}) = \{2-3-4-6\}$$
$$P(F_{13}) = \{3-4-5-6\}$$
$$P(F_{14}) = \{4-5-6\}$$
$$P(F_{15}) = \{5-6\}.$$

The true cost of each path in $P(F_{1i})$ is now found by direct evaluation using the Γ costing function. Some of these true costs have already been found above.

Efficient path	Infimum cost	True cost
1 - 4 - 6	(7,7)	(7,7)
1 - 2 - 3 - 4 - 6	(9,6)	(12,9)
2 - 3 - 4 - 6	(8,5)	(8,5)

So, V_{1i} becomes:

$$V_{11} = \{1-2-3-4-6\}$$
$$V_{12} = \varnothing \Rightarrow P(F_{12}) = \mathrm{Eff}_2(P)$$
$$V_{13} = \varnothing \Rightarrow P(F_{13}) = \mathrm{Eff}_3(P)$$
$$V_{14} = \varnothing \Rightarrow P(F_{14}) = \mathrm{Eff}_4(P)$$
$$V_{15} = \varnothing \Rightarrow P(F_{14}) = \mathrm{Eff}_5(P).$$

Iteration 2

$$u_2 = \{1-2-3-4-6, 1-3-4-5-6, 1-2-3-4-5-6, 2-3-4-5-6\}.$$

So, the avoidance set for this iteration then becomes:

$$S_2 = \{1-2-3-4-5-6,\ 2-3-4-5-6,\ 3-4-5-6,\ 4-5-6,\ 5-6,\ 1-3-4-5-6,\ 1-2-3-4-6,\ 2-3-4-6,\ 3-4-6,\ 4-6\}$$

$$S_{11} = \{1-2-3-4-5-6,\ 1-3-4-5-6,\ 1-2-3-4-6\}$$
$$S_{12} = \{2-3-4-5-6,\ 2-3-4-6\}$$
$$S_{13} = \{3-4-5-6,\ 3-4-6\}$$
$$S_{14} = \{4-5-6,\ 4-6\}$$
$$S_{15} = \{5-6\}.$$

$F_{21}^{(K)}$	Max path-length	Start node	Path	Cost
$F_{21}^{(1)}$	1	1	—	—
$F_{22}^{(1)}$	1	2	—	—
$F_{23}^{(1)}$	1	3	—	—
$F_{21}^{(2)}$	2	1	1 – 4 – 6	(3,4) + (4,3) = (7,7)
$F_{22}^{(2)}$	2	2	—	—
$F_{21}^{(3)}$	3	1	1 – 4 – 6 1 – 3 – 4 – 6	(7,7) (1,4) + (7,4) = (8,8)
$F_{11}^{(4)}$	4	1	1 – 4 – 6	(7,7)
$F_{11}^{(5)}$	5	1	1 – 4 – 6	(7,7)

This phase terminates with the set $F_{2i} = \mathrm{VMIN}\{F_{2i}^{(5)} \cup \{s_{2i}\}\}$.

$$F_{21} = \mathrm{VMIN}\{(7,7),(12,9),(14,12),(10,11)\} = \{(7,7)\}$$
$$F_{12} = \{(8,5)\}$$
$$F_{13} = \{(5,4)\}$$
$$F_{14} = \{(2,3)\}$$
$$F_{12} = \{(1,1)\}$$
$$P(F_{11}) = \{1-4-6\}$$
$$P(F_{12}) = \{2-3-4-6\}$$
$$P(F_{13}) = \{3-4-5-6\}$$
$$P(F_{14}) = \{4-5-6\}$$
$$P(F_{15}) = \{5-6\}.$$

Efficient path	Infimum cost	True cost
1 – 4 – 6	(7,7)	(7,7)

$$V_{21} = \varnothing \Rightarrow P(F_{21}) = \mathrm{Eff}_1(P)$$
$$V_{22} = \varnothing \Rightarrow P(F_{22}) = \mathrm{Eff}_2(P)$$
$$V_{23} = \varnothing \Rightarrow P(F_{23}) = \mathrm{Eff}_3(P)$$
$$V_{24} = \varnothing \Rightarrow P(F_{24}) = \mathrm{Eff}_4(P)$$
$$V_{25} = \varnothing \Rightarrow P(F_{24}) = \mathrm{Eff}_5(P).$$

The algorithm terminates and the following are the non-dominated paths:

$$\begin{aligned} &\mathrm{Eff}_1(P) = \{1-4-6\}, && \mathrm{Eff}_2(P) = \{2-3-4-6\}, \\ &\mathrm{Eff}_3(P) = \{3-4-5-6\}, && \mathrm{Eff}_4(P) = \{4-5-6\}, \\ &\mathrm{Eff}_5(P) = \{5-6\}. \end{aligned}$$

Remark: Path $4-5-6$ would have prevented the path $1-4-6$ from being manifest (since it dominated the subpath $4-6$ of $1-4-6$) had it not been avoided in the backward phases of Iterations 2 and 3.

5. THEORETICAL RESULTS

This section contains the mathematical justification for the algorithm presented in Section 3. All of the results presented here may also have relevance to the foundations of the theory of other related optimization methods and may aid in the derivation of new algorithms.

Note: The notation "$\leq\!\!\neq$" will stand for "does not exceed, vector-wise, but is distinct from". For example, the following notation for vectors C and D: $C \leq D$ and $C \neq D$ could be replaced by $C \leq\!\!\neq D$.

THEOREM 1: *The algorithm terminates after a finite number of iterations.*

Proof: With each iteration, the sets u_r strictly increase. This consideration, and the fact that the number of paths in $P(G)$ is finite, yields the desired result. □

The notation required will now be introduced.

Let $P(G'_r) \equiv P(G) \backslash S_r$, $P_i(G'_r) = P(G'_r) \cap P_i(G)$, $P_i^k(G'_r) = P(G'_r) \cap P_i^k(G)$, and $P_i^{(k)}(G'_r) = P(G'_r) \cap P_i^{(k)}(G)$.

Let $\Pi \in P_i(G'_r)$. Then $\Pi \in \mathrm{Eff}(P_i(G'_r))$ if and only if the set $\{\Pi' : \Pi' \in P_i(G'_r),\ \Gamma_r(\Pi') \leq\!\!\neq \Gamma_r(\Pi)\}$ is empty.

THEOREM 2: $P(F_{ri}^{(n-1)}) = \mathrm{Eff}(P_i(G'_r))$.

Proof: (by induction on k, the cardinality of the path).

Let $k = 1$. Then,

$$\begin{aligned} F_{ri}^{(1)} = &\mathrm{VMIN}\{\Gamma_r(\Pi) : \Gamma_r(\Pi) = \vec{c}_0(i,d), \\ &\text{where } \Pi \in P_i^{(1)}(G) \text{ and } \Pi \notin S_{ri}\}. \end{aligned}$$

Hence, one obtains $P(F_{ri}^{(1)}) = \mathrm{Eff}(P_i^{(1)}(G'_r))$.

Suppose now that $P(F_{ri}^{(k-1)}) = \text{Eff}(P_i^{(k-1)}(G_r'))$, $2 \leq k-1 \leq n$.

A. Let $\Pi \in \text{Eff}(P_i^{(k)}(G_r'))$, where Π is of cardinality k.

Case 1: $\Pi = \{(i,j)\} \cup \Pi_j$, $\Pi_j \in S_{rj}$. Then it follows that $\Gamma_r(\Pi) = \vec{c}_0(i,j) + \Gamma(\Pi_j)$, an element in the set $\{\vec{c}_0(i,j) + \Gamma(\Pi_j), \Pi_j \in S_{rj}^{k-1}\}$. Since $\Pi \in \text{Eff}(P_i^{(k)}(G_r'))$ and has cardinality k, $\Pi \in P(F_{ri}^{(k)})$, as desired.

Case 2: $\Pi = \{(i,j)\} \cup \Pi_j$, $\Pi_j \notin S_r$. We claim that $\Pi_j \in \text{Eff}(P_j^{(k-1)}(G_r'))$. To obtain a proof, suppose not. Then there must exist $\Pi_j' \in P_j^{(k-1)}(G_r')$ such that $\Gamma_r(\Pi_j') \leq\neq \Gamma_r(\Pi_j)$. But then, letting $\Pi' = \{(i,j)\} \cup \Pi_j'$, we get $\Gamma_r(\Pi') \leq\neq \Gamma_r(\Pi)$, by the additivity of Γ_r. This is impossible since $\Pi \in \text{Eff}(P_i^{(k)}(G_r'))$. Hence, $\Pi_j \in \text{Eff}(P_j^{(k-1)}(G_r'))$. By the inductive hypothesis, it follows that $\Pi_j \in P(F_{ri}^{(k-1)})$. Therefore, $\Gamma_r(\Pi) \in \{\vec{c}_0(i,j) + F_{rj}^{(k-1)}\}$. Since $\Pi \in \text{Eff}(P_i^{(k)}(G_r'))$, $\Gamma_r(\Pi) \in F_{ri}^{(k)}$, or $\Pi \in P(F_{ri}^{(k)})$.

B. Suppose now that $\Pi \in P(F_{ri}^{(k)}) \backslash \text{Eff}(P_i^{(k)}(G_r'))$.

This implies the existence of a path $\Pi' \in P(G_r')$ such that $\Gamma_r(\Pi') \leq\neq \Gamma_r(\Pi)$. Without loss of generality, Π' can be assumed to be an element of $\text{Eff}(P_i^{(k)}(G_r'))$. But then, by part A, $\Pi' \in P(F_{ri}^{(k)})$. Since $\Pi \in P(F_{ri}^{(k)})$ by hypothesis, $\Gamma_r(\Pi') \leq\neq \Gamma_r(\Pi)$ is impossible by the definition of $F_{ri}^{(k)}$.

The theorem follows from parts A and B. □

Define the set $\text{Eff}_i(P_r)$ by the condition $\Pi \in \text{Eff}_i(P_r)$ if and only if the set $\{\Pi' : \Pi' \in P_i(G)$ and $\Gamma_r(\Pi') \leq\neq \Gamma_r(\Pi)\}$ is empty.

THEOREM 3: $P(F_{ri}) = \text{Eff}_i(P_r)$.

Proof:

A. Let $\Pi \in \text{Eff}_i(P_r)$.

Case 1: Suppose $\Pi \in P_i(G_r')$. Then, since $\Pi \in \text{Eff}_i(P_r)$, it is also true that $\Pi \in \text{Eff}\,(P_i(G_r'))$; this yields, by Theorem 2, that $\Pi \in P(F_{ri}^{(n-1)})$. But then, since $F_{ri} = \text{VMIN}\{F_{ri}^{(n-1)} \cup \{s_r\}\}$ (and Π is an element of $\text{Eff}_i(P_r)$), we must have $\Pi \in P(F_{ri})$.

Case 2: Suppose $\Pi \notin P_i(G_r')$. Then $\Pi \in S_{ri}$ by definition. Recalling $F_{ri} = \text{VMIN}\{F_{ri}^{(n-1)} \cup \{s_r\}\}$ and since $\Pi \in \text{Eff}_i(P_r)$, Π must be in $P(F_{ri})$.

B. Suppose now that $\Pi \in P(F_{ri}) \backslash \mathrm{Eff}_i(P_r)$. Then there must exist $\Pi' \in P_i(G_r)$ (WLOG $\Pi' \in \mathrm{Eff}_i(P_r)$) such that $\Gamma_r(\Pi') \leq\neq \Gamma_r(\Pi)$. We now identify two cases:

Case 1: Suppose $\Pi' \in S_{ri}$. Then $\Gamma_r(\Pi') \in \{s_{ri}\}$; but, since $\Pi \in P(F_{ri})$, we have that $\Gamma_r(\Pi') \leq\neq \Gamma_r(\Pi)$ is not possible, by the definition of F_{ri}.

Case 2: Suppose $\Pi' \notin S_{ri}$. Then, $\Pi' \in P_i(G'_r)$. Since $\Pi' \in \mathrm{Eff}\,(P_i(G'_r))$, we have, by Theorem 2 that $\Gamma_r(\Pi') \in F_{ri}^{(n-1)}$. But, then $\Gamma_r(\Pi') \leq\neq \Gamma_r(\Pi)$ is impossible since $\Pi \in P(F_{ri})$.

Together parts A and B establish the theorem. □

The next theorem gives the main theoretical result of this paper. It establishes that after a finite number of iterations, the algorithm determines the entire set of efficient paths for all possible initial nodes.

THEOREM 4: *Let the algorithm terminate after t iterations. Then,* $P(F_{ti}) = \mathrm{Eff}_i(P)$.

Proof:

A. Let $\Pi \in \mathrm{Eff}_i(P)$. Let $\Pi_{(t)}$ denote the path Π with cost given by Γ_t.

Case 1: Suppose $\Pi_{(t)} \in u_{ti}$. Note that $V_{ti} = \varnothing$. In the case where $\Pi_{(t)} \in P(F_{ti})$, there is nothing to prove since $\Gamma(\Pi) = \Gamma_t(\Pi_{(t)}) \in F_{ti}$ or $\Pi \in P(F_{ti})$. So, consider the case where $\Pi_{(t)} \notin P(F_{ti})$. Since, by Theorem 3 we have that $P(F_{ti}) = \mathrm{Eff}_i(P_t)$, this implies that $\Pi_{(t)} \in \mathrm{Eff}_i(P_t)$. But, then there must exist a path Π' (WLOG in $\mathrm{Eff}_i(P_t)$), such that $\Gamma_t(\Pi') \leq\neq \Gamma_t(\Pi)$. Since by Theorem 3 $\Pi' \in P(F_{ti})$, and F_{ti} is terminal, $\Gamma_t(\Pi') = \Gamma(\Pi')$. We thus have, $\Gamma(\Pi') = \Gamma_t(\Pi') \leq\neq \Gamma_t(\Pi) = \Gamma(\Pi)$. This is not possible, since $\Pi \in \mathrm{Eff}_i(P)$.

Case 2: Suppose $\Pi_{(t)} \notin u_{ti}$. Once again, if $\Pi_{(t)} \in P(F_{ti})$ there is nothing to prove since t is terminal and $\Gamma(\Pi) = \Gamma_t(\Pi_{(t)})$. So, suppose $\Pi_{(t)} \notin P(F_{ti})$. This implies that there exists Π', (WLOG in $\mathrm{Eff}_i(P_t)$), which, by Theorem 3, is the same as $P(F_{ti})$, such that $\Gamma_t(\Pi') \leq\neq \Gamma_t(\Pi_{(t)})$. But, since $\Gamma_t(\Pi_{(t)}) \leq \Gamma(\Pi_{(t)}) = \Gamma(\Pi)$ ($\Pi_{(t)}$ and Π are the same path), we have $\Gamma_t(\Pi') \leq \Gamma(\Pi)$. However, the facts that $\Gamma_t(\Pi') = \Gamma(\Pi')$ (because $\Pi' \in P(F_{ti})$, t terminal), and $\Gamma_t(\Pi') \leq\neq \Gamma(\Pi)$ from above, force $\Pi \notin \mathrm{Eff}_i(P)$.
Contradiction.

B. Let $\Pi \in P(F_{ti}) \backslash \mathrm{Eff}_i(P)$. This implies that there exists a path $\Pi' \in \mathrm{Eff}_i(P)$ such that $\Gamma(\Pi') \leq\neq \Gamma(\Pi)$. This in turn yields

$\Gamma_t(\Pi') \leq \Gamma(\Pi') \leq\neq \Gamma(\Pi)$. But, since $\Pi \in P(F_{ti})$, $\Gamma_t(\Pi) = \Gamma(\Pi)$; moreover, by Theorem 3, $\Pi \in \mathrm{Eff}_i(P_t)$. These two facts make the inequalities $\Gamma_t(\Pi') \leq \Gamma(\Pi') \leq\neq \Gamma(\Pi)(= \Gamma_t(\Pi))$ impossible.

Parts A and B together establish the theorem. □

A principle of optimality

While the Principle of Optimality of classical dynamic programming is no longer directly applicable to the problem of multi-stage decision optimization with time-dependent cost functions, there is a similar principle that does pertain to this type of problem. It is a generalization of the Principle of Optimality that specializes to classical dynamic programming in the case of time-invariant parameters. This principle requires a rigorous definition of the notion of a partition of a set of paths.

Let (P, Γ) denote a set of paths P with costs determined according to the path-costing function Γ. Let Γ_1 and Γ_2 be two path-costing functions defined on two subsets P_1 and $P_2 = P \backslash P_1$ of P. Then, the set $\{(P_1, \Gamma_1), (P_2, \Gamma_2)\}$ is called a partition of (P, Γ).

Let (P', Γ') denote a set of paths P' costed according to Γ'. A path $\Pi \in P'$ is said to be P'-nondominated if and only if it is nondominated with respect to the set of paths in P' costed according to Γ'.

THEOREM 5 (Principle of Optimality): *Let (P, Γ) be a given set of paths, as in the algorithm. Suppose $\Pi \in P$ is P-nondominated. Then, there exists a unique, finite set of partitions $\{P_r\} = \{\{(P_{1r}, \Gamma_0), (P_2, \Gamma)\}\}$, $P_{1k} \underset{\neq}{\supset} P_{1(k+1)}$ and $P_{2k} \underset{\neq}{\supset} P_{2(k+1)}$, and a nonempty set of non-negative integers I_p such that for at least one $i \in I_p$ every subpath of Π is P_{1i}-nondominated.*

Proof: The algorithm, whose proof appears above, also serves as a constructive proof of this result. An explicit proof follows.

1. **Uniqueness:** This is established by noting that the initial partition is uniquely defined, consisting of all paths costed at their infima, and that subsequent partitions are uniquely determined by the sets F_{ri}, which are themselves unique.
2. **Existence:** Let Π be P-nondominated. Let s be the smallest index for which $\Pi \in P(F_{si}^{(n-1)})$. Such an index exists by the algorithm and finiteness. Suppose now that Π' is a subpath of Π. If Π' coincided with Π, there is nothing to prove, since P is, by virtue of being in $P(F_{si}^{(n-1)})$, P_{1s} nondominated. So, suppose $\Pi' \neq \Pi$. Suppose now that

there exists a path $\Pi'' \in P_{1s}$ such that $\Gamma_0(\Pi'') \neq \Gamma_0(\Pi')$ is true. This yields, $\Gamma_0(\{\Pi \backslash \Pi'\} \cup \{\Pi''\}) \leq \Gamma_0(\{\Pi \backslash \Pi'\} \cup \{\Pi'\}) = \Gamma_0(\Pi)$, and a contradiction. □

Note that this is a generalization of the classical Principle of Optimality since in the case of time invariance the set of partitions has cardinality one, and hence every subpath of a nondominated path must be nondominated.

6. FUTURE RESEARCH

This algorithm was implemented with an object oriented language since data structures (such as arrays, bags, and dictionaries) are useful given the requirements in both the first and second phases of the algorithm to search sets for given subpaths. It is not clear, however, that with increasing problem size this convenience does not carry a price. As the algorithm proceeds, the computational burden shifts from the first phase to the second phase. The time-invariant network of phase one decreases in size while the avoidance set of phase two increases. As such, the burden of computation shifts from the combinatorial operations of dynamic programming to the lookup and comparison operations of vector minimization. It is of interest to investigate how, given efficient data structures (for the information handling required in both phases), the complexity of this algorithm (for optimization with time-varying parameters) compares with dynamic programming in the time-invariant case. The conjecture is that given sufficiently efficient data structures, the two should be comparable, to within a polynomial.

The algorithm, with its alternating dynamic programming and avoidance-set-definition phases has been conceived and formulated as an iterative, region limiting type of algorithm. (It is worth noting that unlike most methods of this type, it is guaranteed to converge to the optimal solution in a finite number of iterations.) Recent work by Getachew [11] suggests that a fruitful re-formulation exists in terms of *stratified* partially ordered sets. A POSET $\wp$ is said to be stratified if there exists a sequence of functions $\{c_j\}_{j \in J}$, $c_j : \wp \to \mathfrak{I}$, $\mathfrak{I}$ a POSET, such that for each $p \in \wp$ the sequence $\{c_j(p)\}$ is monotonic in j. Such an abstract conception has made it possible to see that the algorithm is applicable to, among others, fuzzy multi-stage decision problems where the constraint and goal sets are time-varying.

ACKNOWLEDGEMENTS

This research was supported in part by Grant 60NANBOD1023 from the Building and Fire Research Laboratory, National Institute of Standards and Technology, Gaithersburg, Maryland.

REFERENCES

1. R. E. BELLMAN, *On a Routing Problem*, Quarterly Appl. Math., 1958, *16*, p. 87-90.
2. T. A. BROWN and R. E. STRAUCH, *Dynamic Programming in Multiplicative Lattices*, J. Math. Anal. Appl., 1965, *12*, p. 364-370.
3. J. BRUMBAUGH-SMITH and D. SHIER, *An empirical investigation of some bicriterion shortest path algorithms*, European J. Op. Res., 1989, *43*, p. 216-224.
4. R. L. CARRAWAY and T. L. MORIN, *Generalized Dynamic Programming for Multicriteria Optimization*, European J. Op. Res., 1990, *44*, p. 95-104.
5. K. L. COOKE and E. HALSEY, *The Shortest Route Through a Network with Time-Dependent Internodal Transit Times*, J. Math. Anal. Appl., 1966, *14*, p. 493-498.
6. H. W. CORLEY and I. D. MOON, *Shortest Paths in Networks with Vector Weights*, J. Opt. Theory Appl., 1985, *46*, p. 79-86.
7. 7. H. G. DAELLENBACH and C. A. DEKLUYVER, *Note on Multiple Objective Dynamic Programming*, J. Op. Res. Soc., 1980, *31*, p. 591-594.
8. E. W. DIJKSTRA, *A Note on Two Problems in Connection with Graphs*, Num. Math., 1959, *1*, p. 269-271.
9. S. E. DREYFUS, *An Appraisal of Some Shortest Path Algorithms*, Ops. Res., 1969, *17*, p. 395-412.
10. T. GETACHEW, *An Algorithm for Multiple-Objective Network Optimization with Time Variant Link-Costs*, Ph.D. dissertation, Clemson Univ., Clemson, South Carolina, USA, 1992.
11. T. GETACHEW, *Optimization over Stratified Posets*, in preparation.
12. J. HALPERN, *Shortest Route with Time-dependent Length of Edges and Limited Delay Possibilities in Nodes*, J. Ops. Res., 1977, *21*, p. 117-124.
13. M. I. HENIG, *The Principle of Optimality in Dynamic Programming with Returns in Partially Ordered Sets*, Math. Ops. Res., 1985, *10*, p. 462-470.
14. D. E. KAUFMANN and R. L. SMITH, *Minimum Travel Time Paths in Dynamic Networks with Application to Intelligent Vehicle-highway Systems*, University of Michigan, Transportation Research Institute, Ann Arbor, Michigan, USA, IVHS Tech. Rpt. 90-11, 1990.
15. M. M. KOSTREVA and M. M. WIECEK, *Time Dependency in Multiple Objective Dynamic Programming*, J. Math. Anal. Appl., 1993, *173*, p. 289-308.
16. A. ORDA and R. ROM, *Shortest-path and Minimum-delay Algorithms in Networks with Time-dependent Edge-length*, J. Assoc. Comp. Mach., 1990, *37*, p. 607-625.
17. A. B. PHILPOTT, *Continuous-time Shortest Path Problems and Linear Programming*, SIAM J. Cont. Opt., 1994, *32*, p. 538-552.
18. S. VERDU and H. V. POOR, *Abstract Dynamic Programming Models under Commutativity Conditions*, SIAM J. Cont. Opt., 1987, *25*, p. 990-1006.