RAIRO Operations Research
RAIRO Oper. Res. **36** (2002) 175-190
DOI: 10.1051/ro:2003001

A DIMENSION-REDUCTION ALGORITHM FOR MULTI-STAGE DECISION PROBLEMS WITH RETURNS IN A PARTIALLY ORDERED SET

Teodros Getachew[1] and Michael M. Kostreva[2]

Communicated by Franco Giannessi

Abstract. In this paper a two-stage algorithm for finding non-dominated subsets of partially ordered sets is established. A connection is then made with dimension reduction in time-dependent dynamic programming *via* the notion of a bounding label, a function that bounds the state-transition cost functions. In this context, the computational burden is partitioned between a time-independent dynamic programming step carried out on the bounding label and a direct evaluation carried out on a subset of "real" valued decisions. A computational application to time-dependent fuzzy dynamic programming is presented.

Keywords. Multi-criteria optimization, time-variant networks, dimension reduction.

1. Introduction

Dynamic Programming, as formulated by Bellman [1], has proven to be one of the most fruitful contributions to the solution of multi-stage decision problems. Over the years, it has been generalized to decision problems with returns in multiplicative lattices by Brown and Strauch [3], and partially ordered sets by Henig [12]. The domain of application of this principle has also expanded to

Received December, 2000.

[1] Department of Management, Providence College, Providence, RI 02918-0001, U.S.A.
[2] Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-1907, U.S.A.

cover decision problems involving multiple objectives in works by Daellenbach and DeKluyver [6] and Corley and Moon [5].

Cooke and Halsey [4] proposed one of the earliest algorithms for Dynamic Programming in a time-dependent context. Their application concerned a routing problem with time-dependent transition times between nodes in a network where the temporal dimensions was incorporated into the state of the system *via* a time-grid. Other work in this vein includes Dreyfus [8] who proposed a modification of Djikstra's [7] famous shortest path algorithm to finding the shortest paths in networks with time-dependent arc lengths and Halpern [11] and Orda and Rom [15] who considered a similar problem with waiting ("parking") allowed at nodes. Kostreva and Wiecek [14] introduced two algorithms that unified the themes of multi-objectivity with time-dependency. Their first algorithm extended Cooke and Halsey's result to the multi-objective case, while their second algorithm generalized earlier work by Kaufman and Smith [13] (on finding minimum travel time paths in networks with time-varying transit times), under the restriction that all cost functions be non-decreasing. Contributions which connect fuzzy sets and dynamic programming are reviewed in Kacprzyk and Esogbue [16] and a multicriteria fuzzy problem is handled by scalarization in the paper of Hussein and Abo–Sinna [17]. Finally, Getachew *et al.* [10] proposed a "backward-forward", recursive algorithm for finding all the non-dominated paths in a network given time-dependent cost functions, the only restriction on the cost functions being boundedness.

In this paper we will show the result in Getachew *et al.* [10] to be a particular instance of an algorithm for finding non-dominated subsets of partially ordered sets satisfying a simple "bounded label" condition. We shall then show how this general algorithm can be applied to decision making in a fuzzy multiple objective context with time dependency. Thus, the new approach will comprehend more complex and more realistic models of decision making than any of the above previous research contributions.

2. Non-dominated subsets of partially ordered sets

2.1. Notation, definitions and terminology

Definition 1. Let $\mathbb{S}$ and $\mathbb{P}$ be two sets, $\mathbb{P}$ partially ordered, with partial order "$\geqq$". Let $\mathcal{P}$ be any nonempty subset of $\mathbb{S}$. Let $\mathfrak{E}$: $\mathcal{P} \to \mathbb{P}$ and $\mathcal{C}$: $\mathcal{P} \to \mathbb{P}$. For any $p \in \mathcal{P}$, $\mathcal{C}(p)$ will be called the $\mathcal{C}$-cost of p, and $\mathfrak{E}(p)$ the $\mathfrak{E}$-cost of p.

Definition 2. Let p, $q \in \mathcal{P}$, $\mathcal{P}$ as defined above.

(1) If $\mathfrak{E}(p) \geqq \mathfrak{E}(q)$ (or $\mathfrak{E}(q) \geqq \mathfrak{E}(p)$ then p and q are said to be $\mathfrak{E}$-comparable. ($\mathcal{C}$-comparability is defined analogously).
(2) If $\mathfrak{E}(p) \geqq \mathfrak{E}(q)$ and $\mathfrak{E}(p) \neq \mathfrak{E}(q)$, p is said to $\mathfrak{E}$-dominate q. This will be denoted as $\mathfrak{E}(p) > \mathfrak{E}(q)$ ($\mathcal{C}$-domination (and notation) is defined analogously).

(3) Let $\mathcal{Q} \subseteq \mathcal{P}$ and $p \in \mathcal{P}$. If the set $\{q \in \mathcal{Q}\colon\ q\ \mathfrak{E}\text{-dominates } p\}$ is empty, then p is said to be $\mathfrak{E}$-nondominated over $\mathcal{Q}$; if $\mathcal{Q} = \mathcal{P}$, then p will simply be said to be $\mathfrak{E}$-nondominated. (These notions are defined analogously for $\mathcal{C}$).

Notation 1. Let $\mathcal{P}$ be as defined above. By the main decision problem (MDP) we will mean the problem: find all $\mathfrak{E}$-nondominated elements of $\mathcal{P}$.

Definition 3. Let $\mathcal{P}$, $\mathcal{C}$ and $\mathfrak{E}$ be as defined above. If for all $p \in \mathcal{P}$, $\mathcal{C}(p) \geqq \mathfrak{E}(p)$, then $\mathcal{C}$ is said to be a Uniformly (upper) Bounding Label for $\mathfrak{E}$.

Notation 2. Let $\mathcal{Q}, \mathcal{R} \subseteq \mathcal{P}$, $\mathcal{P}$ a set as defined above.

(a) $[\mathcal{Q}]_{\mathfrak{E}} \equiv \{(q, \mathfrak{E}(q))\colon q \in \mathcal{Q}\}$;
(b) $[\mathcal{Q}]_{\mathfrak{E}1} \equiv \{q\colon (q, \mathfrak{E}(q)) \in [\mathcal{Q}]_{\mathfrak{E}}\}$;
(c) $\mathcal{ND}[[\mathcal{Q}]_{\mathfrak{E}}] \equiv \{(q, \mathfrak{E}(q)) \in [\mathcal{Q}]_{\mathfrak{E}}\colon\ q \text{ is } \mathfrak{E}\text{-nondominated}\}$;
(d) $\mathcal{ND}[[\mathcal{Q}]_{\mathfrak{E}}]_1 \equiv \{q\colon (q, \mathfrak{E}(q)) \in \mathcal{ND}[[\mathcal{Q}]_{\mathfrak{E}}]\}$;
(In light of the notation above, MDP can now be restated as: find $\mathcal{ND}[[\mathcal{P}]_{\mathfrak{E}}]_1$.)
(e) $\mathcal{ND}[[\mathcal{Q}]_{\mathfrak{E}}]_2 \equiv \{\mathfrak{E}(q)\colon (q, \mathfrak{E}(q)) \in \mathcal{ND}[[\mathcal{Q}]_{\mathfrak{E}}]\}$;
With analogous notation holding for $\mathcal{C}$, we finally have
(f) $\mathcal{ND}[[\mathcal{Q}]_{\mathfrak{E}} \cup [\mathcal{R}]_{\mathcal{C}}] \equiv \{\{q \in \mathcal{ND}[[\mathcal{Q}]_{\mathfrak{E}}]_1\colon \mathcal{C}(r) \geqq \mathfrak{E}(q) \text{ and } \mathcal{C}(r) \neq \mathfrak{E}(q) \text{ is false for all } r \in \mathcal{R}\} \cup \{r \in \mathcal{ND}[[\mathcal{R}]_{\mathcal{C}}]_1\colon\ \mathfrak{E}(q) \geqq \mathcal{C}(r) \text{ and } \mathfrak{E}(q) \neq \mathcal{C}(r) \text{ is false for all } q \in \mathcal{Q}\}$.

2.2. Algorithm

The first iteration of the algorithm begins by determining all $\mathcal{C}$-nondominated elements of $\mathcal{P}$; this computation creates the set $\mathcal{N}_1$. For this iteration this will coincide with the set $\mathcal{T}_1$ (since the set $\mathcal{A}_1$ is empty). For each element of this set $\mathcal{T}_1$, the corresponding $\mathfrak{E}$-value is computed. The set of all those elements in this set whose $\mathcal{C}$-value is different from their $\mathfrak{E}$-value forms the set $\mathcal{A}_2$. If $\mathcal{A}_2$ is empty, the algorithm stops; $\mathcal{T}_1$ solves the Main Decision Problem. Otherwise, the second iteration of the algorithm commences with the determination of all $\mathcal{C}$-nondominated elements in the complement of $\mathcal{A}_1 \cup \mathcal{A}_2$. This operation yields the set $\mathcal{N}_2$. It then determines the nondominated set of the set $\mathcal{N}_2 \cup (\mathcal{A}_1 \cup \mathcal{A}_2)$ with the elements of $\mathcal{N}_2$ having their $\mathcal{C}$-value and the elements of $(\mathcal{A}_1 \cup \mathcal{A}_2)$ their $\mathfrak{E}$-value. The nondominated elements so determined constitute the set $\mathcal{T}_2$. As in the first iteration, the set $\mathcal{A}_3$ consisting of all those elements in $\mathcal{T}_2$ whose $\mathcal{C}$-value is different from their $\mathfrak{E}$-value is determined by direct evaluation. If this set is empty, the algorithm halts; the set $\mathcal{T}_2$ solves the Main Decision Problem. Otherwise, the next iteration commences by determining the set of all $\mathcal{C}$-nondominated elements in the complement of $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$, and so on until convergence (to the solution of the Main Decision Problem).

The algorithm is presented formally next.

Let $\mathcal{P}$, $\mathcal{C}$ and $\mathfrak{E}$ be as defined above, $\mathcal{P}$ a finite set. Consider the following algorithm $\mathcal{G}$.

BEGIN

0: $\mathcal{A}_1 := \emptyset$; $k := 1$;

DO

1: $\mathcal{N}_k := \mathcal{ND}\left[\left[\mathcal{P}\setminus \bigcup_{i=1}^{k} \mathcal{A}_i\right]_{\mathcal{C}}\right]_1$;

2: $\mathcal{T}_k := \mathcal{ND}\left[[\mathcal{N}_k]_{\mathcal{C}} \cup \left[\bigcup_{i=1}^{k} \mathcal{A}_i\right]_{\mathfrak{E}}\right]_1$;

3: $\mathcal{A}_{k+1} := \{p \in \mathcal{T}_k : \mathcal{C}(p) \neq \mathfrak{E}(p)\}$;

4: $k := k + 1$;

WHILE $(\mathcal{A}_{k+1} \neq \emptyset)$

END

The following diagram summarizes the Algorithm.

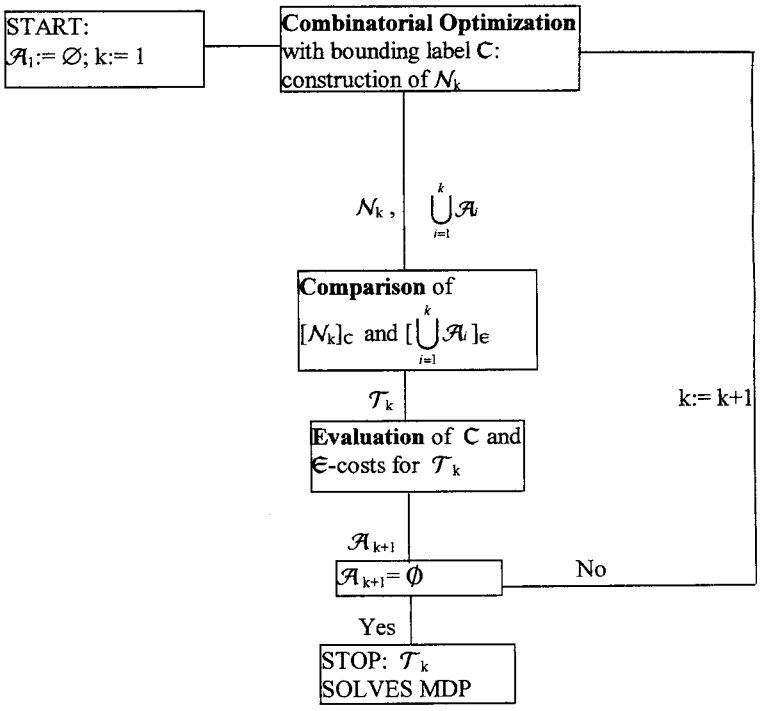

FIGURE 1

Note the computational partitioning: combinatorial considerations enter only in the first phase of the algorithm and only involve the bounding label $\mathcal{C}$. All subsequent steps involve either comparison or direct evaluation, using already constructed elements. In particular, consideration of the "real" $\mathfrak{E}$-costs is confined to these last two steps.

Therefore, given the ambient structure that characterizes this algorithm, consisting of costs $\mathcal{C}$, $\mathfrak{E}$: $\mathcal{P} \to \mathbb{P}$, $\mathbb{P}$ a partially ordered set and $\mathcal{P}$ a finite (nonempty) set, the task of solving MDP, in as much as it involves combinatorial considerations, may benefit from the introduction of a partitioning such as is suggested above, *provided that the bounding label $\mathcal{C}$ allows for a reduction of combinatorial complexity*[3].

We are now ready to state and prove the main theoretical result of this paper.

Proposition 1. Let $\mathcal{P}$, $\mathcal{C}$ and $\mathfrak{E}$ be as defined above, $\mathcal{P}$ a finite set. If $\mathcal{C}$ is a Uniformly Bounding Label for $\mathfrak{E}$, then algorithm $\mathcal{G}$ converges to a solution of the Main Decision Problem in a finite number of iterations.

Proof.

(i) Finiteness.
Suppose $\mathcal{G}$ does not terminate in a finite number of iterations. Let $N > 0$. We must then have that for each i, $1 \leq i \leq N$, $|\mathcal{A}_i| \geq 1$. Now, since $\mathcal{A}_{k+1} \subseteq \mathcal{N}_k$ and $\mathcal{N}_k \cap \bigcup_{i=1}^{k} \mathcal{A}_i = \emptyset$, the $\mathcal{A}_i$'s must be pairwise disjoint. But then, $\bigcup_{i=1}^{N} \mathcal{A}_i \subseteq \mathcal{P}$ implies that $|\mathcal{P}| \geq N$. Since N was arbitrary, $\mathcal{P}$ cannot be a finite set. Contradiction.

(ii) Convergence.
Suppose $p \in \mathcal{ND}[[\mathcal{P}]_{\mathfrak{E}}]_1 \backslash \mathcal{T}_t$. Consider the following cases:

a) $p \in \mathcal{N}_t$ (and $p \notin \mathcal{T}_t$).
Then there exists q, $q \in \bigcup_{i=1}^{t} \mathcal{A}_i$, $q \neq p$ and $\mathfrak{E}(q) > \mathcal{C}(p)$. But then, since $\mathcal{C}(p) \geqq \mathfrak{E}(p)$, q $\mathfrak{E}$-dominates p, contradicting $p \in \mathcal{ND}[[\mathcal{P}]_{\mathfrak{E}}]_1$.

b) $p \notin \mathcal{N}_t$ (and $p \notin \mathcal{T}_t$). Consider the following sub-cases.

(1) $p \in \bigcup_{i=1}^{t} \mathcal{A}_i$ (and $p \notin \mathcal{T}_t$). This implies that there exists q, $q \in \mathcal{N}_t \cup \bigcup_{i=1}^{t} \mathcal{A}_i$ such that either, $\mathcal{C}(q) > \mathfrak{E}(p)$ (if $q \in \mathcal{N}_t$) or $\mathfrak{E}(q) > \mathfrak{E}(p)$ (if $q \in \bigcup_{i=1}^{t} \mathcal{A}_i$) is true. The latter case renders $p \in \mathcal{ND}[[\mathcal{P}]_{\mathfrak{E}}]_1$ impossible directly. In the former case, since without loss of generality $q \in \mathcal{T}_t$ and t is the terminal iteration, $\mathcal{C}(q) = \mathfrak{E}(q)$ again forcing a contradiction by rendering $p \in \mathcal{ND}[[\mathcal{P}]_{\mathfrak{E}}]_1$ impossible.

[3]This is not to say that the contributions of comparison and evaluation to the total computational cost can be ignored; however, the relative efficiency of search algorithms and the plummeting cost of storage would make this cost less onerous by comparison.

(2) $p \in \mathcal{P} \setminus \bigcup_{i=1}^{t} \mathcal{A}_i$. Then there exists q, $q \in \mathcal{P} \setminus \bigcup_{i=1}^{t} \mathcal{A}_i$ such that $\mathcal{C}(q) > \mathfrak{E}(p)$. Without loss of generality, we can pick this $q \in \mathcal{N}_t$ and indeed $\mathcal{T}_t$. But then, since t is the terminal iteration, $\mathcal{C}(q) = \mathfrak{E}(q)$, implying that q $\mathfrak{E}$-dominates p, contradicting $p \in \mathcal{ND}[[\mathcal{P}]_{\mathfrak{E}}]_1$.

Whence $\mathcal{ND}[[\mathcal{P}]_{\mathfrak{E}}]_1 \subseteq \mathcal{T}_t$.

Suppose on the other hand there exists p, $p \in \mathcal{T}_t \setminus \in \mathcal{ND}[[\mathcal{P}]_{\mathfrak{E}}]_1$.
Then there exists q, such that $\mathfrak{E}(q) > \mathfrak{E}(p)$. Consider the following cases:

(1) $q \in \bigcup_{i=1}^{t} \mathcal{A}_i$.

Suppose now that $p \in \bigcup_{i=1}^{j} \mathcal{A}_i$ for some j. Then, $p \in \mathcal{T}_t$ would not be possible since by the terminal iteration, q will have $\mathfrak{E}$-dominated p. Therefore, $p \in \bigcup_{i=1}^{j} \mathcal{A}_i$ is impossible for any j. Therefore we must have $p \in \mathcal{P} \setminus \bigcup_{i=1}^{t} \mathcal{A}_i$. Since $p \in \mathcal{T}_t$, $p \in \mathcal{N}_t$ must thus be true. But then, $\mathfrak{E}(q) > \mathcal{C}(p)$ is impossible (for otherwise p could not be in $\mathcal{T}_t$, being dominated by an element of $\bigcup_{i=1}^{t} \mathcal{A}_i$, namely q). By terminality of t, $\mathfrak{E}(p) = \mathcal{C}(p)$, rendering $\mathfrak{E}(q) > \mathfrak{E}(p)$ also.

(2) $q \notin \bigcup_{i=1}^{t} \mathcal{A}_i$. Consider now the two sub-cases.

1. $p \in \bigcup_{i=1}^{t} \mathcal{A}_i$. Since $q \in \mathcal{P} \setminus \bigcup_{i=1}^{t} \mathcal{A}_i$, we must have $q \in \mathcal{N}_t$. Otherwise, there must exist an r, $r \in \mathcal{P} \setminus \bigcup_{i=1}^{t} \mathcal{A}_i$, such that $\mathcal{C}(r) > \mathcal{C}(q)$. Without loss of generality, we can assume that $r \in \mathcal{N}_t$. Since $\mathfrak{E}(q) > \mathfrak{E}(p)$ and $\mathcal{C}(r) > \mathcal{C}(q) \geqq \mathfrak{E}(q)$, we must have $\mathcal{C}(r) > \mathfrak{E}(p)$, rendering $p \in \mathcal{T}_t$ impossible. So, $q \in \mathcal{N}_t$ (without loss of generality, we can assume that $q \in \mathcal{T}_t$) and so $\mathcal{C}(q) = \mathfrak{E}(q)$ by terminality of t. This forces $\mathcal{C}(q) > \mathfrak{E}(p)$, thus rendering $p \in \mathcal{T}_t$ impossible.
2. $p \notin \bigcup_{i=1}^{t} \mathcal{A}_i$, or $p \in \mathcal{P} \setminus \bigcup_{i=1}^{t} \mathcal{A}_i$. Since $p \in \mathcal{T}_t$, $p \in \mathcal{N}_t$ and by terminality $\mathcal{C}(p) = \mathfrak{E}(p)$. Now, on the other hand, since $\mathfrak{E}(q) > \mathfrak{E}(p)$, and without loss of generality $q \in \mathcal{N}_t$ we have $\mathcal{C}(q) \geqq \mathfrak{E}(q) > \mathfrak{E}(p) = \mathcal{C}(p)$ making $p \in \mathcal{T}_t$ impossible.

Whence $\mathcal{T}_t \subseteq \mathcal{ND}[[\mathcal{P}]_{\mathfrak{E}}]_1$.

3. Application to time-dependent dynamic programming

The application we present next suggests that the class of time-dependent multi-stage decision problems does admit of such a fruitful partitioning; that is, the combinatorial optimization step, in this case dynamic programming, can be carried out with a bounding label $\mathcal{C}$ of lesser dimension than would have been associated with "real" cost $\mathfrak{E}$.

Our formulation is a extension of a general Fuzzy Decision Problem formulated by Bellman and Zadeh [2]. In this formulation, the system under control is deterministic, with states σ_i varying over a finite state space X, and a set of inputs α_j from an input space U. State-Transition is given by a function f: $X \times U \to X$ which depends on the state and the input u_i at stage i; that is,

$$x_{i+1} = f(x_i, u_i),\ i = 1, 2, ...N-1.$$

For each stage n, there are two types of constraints, characterized by membership functions:

(i) corresponding to the stage k, a fuzzy constraint with membership function μ_k which is a function of the input input u_i;

(ii) corresponding to the goal (at the terminal stage N), a fuzzy constraint with membership function μ_G^N which is a function of the state at the terminal stage σ_i.

A decision $D \equiv (u_0, u_1..., u_{N-1})$ is then characterized by a "confluence" membership function μ_D given by:

$$\mu_D(u_0, u_1..., u_{N-1}) \equiv \mu_0(u_0) \wedge \mu_1(u_1) \wedge ... \wedge \mu_{N-1}(u_{N-1}) \wedge \mu_G^N(x_N),$$

where x_N is obtained from u_0, u_1..., u_{N-1} and the initial state x_0, by iterations of f.

The decision problem then becomes: find those decisions D with maximal (non-dominated) values for confluence functions costs μ_D. This is readily done *via* the recurrence equations,

$$\mu_G^{N-\nu}(x_{N-\nu}) = \mathrm{Max}_u^{N-\nu}(\mu_{N-\nu}(u_{N-\nu}) \wedge \mu_G^{N-\nu+1}(x_{N-\nu+1})),$$

$$x_{N-\nu+1} = f(x_{N-\nu},\, u_{N-\nu}), \quad \nu = 1, ..., N.$$

We now modify this problem to include time-dependency, as follows.

Let there be a state-transition-time function T: $X \times X \times U \to \mathbb{R}_+ \cup \{0\}$, that gives the time it takes to enter a state $x_i \in X$ from some other state $x_j \in X$ *via* an input $u_k \in U$. Similarly, let the fuzzy constraint functions u_i: $U \times \mathbb{R}_+ \cup \{0\} \to \mathbb{R}_+ \cup \{0\}$ be functions of input $u_k \in U$ and time $t(\equiv t(u_k))$ at input. Finally, let the membership function characterizing the fuzzy constraint on the goal, μ_G^N: $X \times \mathbb{R}_+ \cup \{0\} \to \mathbb{R}_+ \cup \{0\}$, be a function of the terminal state x_N and the time $t(\equiv t(u_G^N))$ of arrival at this final state[4]. In this modified problem, a decision

[4]In this paper, we shall take "times of arrival" and "times of input" to be crisp notions.

takes the form,

$$D \equiv (u_0, t(u_0)),\ (u_1, t(u_1)), ..., (u_{N-1}, t(u_{N-1})),$$

where $t(u_i)$ is the time of input u_i.

With each such decision D, we shall associate the confluence function

$$\mu_D \equiv \mu_0(u_0, t(u_0)) \wedge \mu_1(u_1, t(u_1)) \wedge ... \wedge \mu_{N-1}(u_{N-1}, t(u_{N-1})) \wedge \mu_G^N(x_N, t(x_N)).$$

Once again, the problem is:
Find all decisions D with maximal (or non-dominated) values for confluence functions.
As noted in the introduction to this section, the standard approach to solving this second problem would be to somehow incorporate time into the state space of the system and by invoking the principle of optimality derive and solve the appropriate recurrence equations. Cooke and Halsey [4], when they first considered Dynamic Programming in a time-dependent context, took the time of arrival at a node in a network to be part of the state characterizing the network, capturing this enlarged state space by a grid enumeration of all possible arrival times. While this was in the spirit of the concept of state for multi-stage decision problems (see for example ElMagrahby [9]), problems involving large networks with time-dependent internodal transition times, and multi-criteria objective functions make this approach computationally prohibitive, due chiefly to the exponential growth of the size of the state space.

The motivation behind the algorithm that we propose is the observation that this enlargement of the state space can be partially mitigated by replacing a one step algorithm over the entire state space (which includes time) by a backward-forward algorithm with time being excluded from the backward step and instead "evaluated-out" in the forward step over a subset of the underlying decision space.

Prior to starting with the computations, we shall first make the following notational adjustments:

$\mathbb{P} \equiv (\mathbb{P}, \geqq) \equiv (\mathbb{R} \times \mathbb{R}, \geqq)$, where "$\geqq$" is component-wise comparison of two-dimensional vectors and $\mathbb{R}$ is the set of real numbers with the usual order.

$\mathcal{P} \equiv \{D \equiv D((u_0, t(u_0)), (u_1, t(u_1)), ..., (u_{N-1}, t(u_{N-1}))) \colon\ u_i \in U\}$

$\mathcal{C}$: $\mathcal{P} \to \mathbb{P}$ given by,

$$\mathcal{C}(D) = \sup_t \mu_0(u_0, t) \wedge \sup_t \mu_1(u_1, t) \wedge ... \wedge \sup_t \mu_{N-1}(u_{N-1}, t) \wedge \sup_t \mu_G^N(x_N, t),$$
$$t \in \mathbb{R}_+ \cup \{0\}$$

$\mathfrak{E}$: $\mathcal{P} \to \mathbb{P}$ given by,

$$\mathfrak{E}(D) = \mu_0(u_0, t(u_0)) \wedge \mu_1(u_1, t(u_1)) \wedge ... \wedge \mu_{N-1}(u_{N-1}, t(u_{N-1})) \wedge \mu_G^N(x_N, t(x_N)),$$

where, given an initial state x_0, and letting $t(u_0) = 0$, the subsequent sequence of states x_1, x_2,..., x_N induced by the inputs u_0, u_1, u_2,..., u_{N-1}, $t(u_k)$ is evaluated by:

$$t(u_k) = \sum_{i=0}^{k-1} T(x_i, x_{i+1}, u_i).$$

We can now readily observe that

(1) $\mathcal{P}$ is finite, and
(2) $\mathcal{C}$ is a bounding label for $\mathfrak{E}$.

This enables us to invoke the main proposition to solve the problem.

The algorithm has a "backward" and "forward" phase. Note in particular that the backward phase is carried out with *time-independent* costs (the $\sup_t \mu_i$'s), thus achieving the reduction in dimensionality, as desired.

4. A numerical example

We next present a numerical example in which we carry out the algorithmic computations in detail.

This is the data for the example:

a state space $X = \{\sigma_1, \sigma_2, \text{ and } \sigma_3\}$;
an input space $U = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6\}$;
a state transition function f given by the following matrix:

x_i / u_i	σ_1	σ_2	σ_3
α_1	σ_1	σ_2	σ_3
α_2	σ_1	σ_3	σ_2
α_3	σ_2	σ_1	σ_3
α_4	σ_2	σ_3	σ_1
α_5	σ_3	σ_1	σ_2
α_6	σ_3	σ_2	σ_1

A state-to-state transition time function T given by:

u_i / $\sigma_j \to \sigma_k$	α_1	α_2	α_3	α_4	α_5	α_6
$\sigma_1 \to \sigma_1$	0	0				
$\sigma_1 \to \sigma_2$			6	2		
$\sigma_1 \to \sigma_3$					3	5
$\sigma_2 \to \sigma_2$	0					0
$\sigma_2 \to \sigma_1$			7		3	
$\sigma_2 \to \sigma_3$		2		1		
$\sigma_3 \to \sigma_3$	0		0			
$\sigma_3 \to \sigma_1$				1		5
$\sigma_3 \to \sigma_2$		2			5	

Multi-Criteria Constraint Membership Functions:

$$\mu_k(\alpha_j, t) \equiv \begin{cases} \mu_{k1}(\alpha_i, t) \\ \mu_{k2}(\alpha_i, t) \end{cases} \quad \text{for } k = 0, 1;$$

	α_1	α_2	α_3	α_4	α_5	α_6
$\mu_0(\alpha_j, t)$	$\begin{cases} 0.0 \\ 0.1 \end{cases}$	$\begin{cases} 0.3 \\ 0.8 \end{cases}$	$\begin{cases} 0.3 \; if \; t < 3 \; and \; 0.7 \; otherwise \\ 0.8 \; if \; t < 2 \; and \; 0.3 \; otherwise \end{cases}$	$\begin{cases} 0.5 \\ 0.5 \end{cases}$	$\begin{cases} 0.3 \\ 0.7 \end{cases}$	$\begin{cases} 0.2 \\ 0.5 \end{cases}$

	α_1	α_2	α_3	α_4	α_5	α_6
$\mu_1(\alpha_j, t)$	$\begin{cases} 0.9 \\ 0.1 \end{cases}$	$\begin{cases} 0.2 \; if \; t < 2 \\ and \; 0.8 \; otherwise \\ 0.3 \end{cases}$	$\begin{cases} 0.5 \\ 0.9 \end{cases}$	$\begin{cases} 0.4 \\ 0.6 \; if \; t < 5 \\ and \; 0.2 \; otherwise \end{cases}$	$\begin{cases} 0.1 \\ 0.1 \end{cases}$	$\begin{cases} 0.7 \\ 0.3 \end{cases}$

and the goal membership function for the terminal stage $N = 2$,

$$\mu_G^2(x_2, t) \equiv \begin{cases} \mu_{G^2 1}(x_2, t) \\ \mu_{G^2 2}(x_2, t) \end{cases} \quad \text{for } x_2 \in \{\sigma_1, \sigma_2, \sigma_3\}.$$

	σ_1	σ_2	σ_3
$\mu_G^2(x_2, t)$	$\begin{cases} 0.8 \\ 0.7 \end{cases}$	$\begin{cases} 0.5 \\ 0.8 \end{cases}$	$\begin{cases} 0.5 \; if \; t < 8 \; and \; 0.9 \; otherwise \\ 0.4 \; if \; t < 7 \; and \; 0.9 \; otherwise \end{cases}$

We are now ready to carry out the computations leading to the solution of the MDP. Find all the $\mathfrak{E}$-nondominated decisions in the system depicted above[5]. We find it convenient to organize out calculations in tabular form.

α_j	$f(\sigma_1, \alpha_j)$	$\sup_t \mu_1(\alpha_j)$	$\sup_t \mu_G^2(\alpha_j, \sigma_1)$	$\sup_t \mu_1(\alpha_j) \wedge \sup_t \mu_G^2(\alpha_j, \sigma_1)$
α_1	σ_1	(0.9, 0.1)	(0.8, 0.7)	(0.8, 0.1)
α_2	σ_1	(0.8, 0.3)	(0.8, 0.7)	(0.8, 0.3)*
α_3	σ_2	(0.5, 0.9)	(0.3, 0.8)	(0.5, 0.8)*
α_4	σ_2	(0.4, 0.6)	(0.3, 0.8)	(0.3, 0.6)
α_5	σ_3	(0.1, 0.1)	(0.9, 0.9)	(0.1, 0.1)
α_6	σ_3	(0.7, 0.3)	(0.9, 0.9)	(0.7, 0.3)

* $\mu_G^1(\sigma_1)$

[5]For the backward (constant cost) phase we shall still be using the recurrence relations,

$$\mu_G^{N-\nu}(x_{N-\nu}) = \mathrm{Max}_u^{N-\nu} \mu_{N-\nu}(u_{N-\nu}) \wedge \mu_G^{N-\nu+1}(x_{N-\nu+1}),$$
$$x_{N-\nu+1} = f(x_{N-\nu}, u_{N-\nu}), \; \nu = 1, ..., N$$

where "Max" will be replaced by "Vector-(Component-Wise) Max", "$\mu_{N-\nu}$" by "$\sup_t \mu_{N-\nu}$" and "$\mu_G^{N-\nu+1}$" by "$\sup_t \mu_G^{N-\nu+1}$".

α_j	$f(\sigma_2, \alpha_j)$	$\sup_t \mu_1(\alpha_j)$	$\sup_t \mu^2_G(\alpha_j, \sigma_2)$	$\sup_t \mu_1(\alpha_j) \wedge \sup_t \mu^2_G(\alpha_j, \sigma_2)$
α_1	σ_2	(0.9, 0.1)	(0.3, 0.8)	(0.3, 0.1)
α_2	σ_3	(0.8, 0.3)	(0.9, 0.9)	(0.8, 0.3)*
α_3	σ_1	(0.5, 0.9)	(0.8, 0.7)	(0.5, 0.7)*
α_4	σ_3	(0.4, 0.6)	(0.9, 0.9)	(0.4, 0.6)
α_5	σ_1	(0.1, 0.1)	(0.8, 0.7)	(0.1, 0.1)
α_6	σ_2	(0.7, 0.3)	(0.3, 0.8)	(0.3, 0.3)

* $\mu^1_G(\sigma_2)$

α_j	$f(\sigma_3, \alpha_j)$	$\sup_t \mu_1(\alpha_j)$	$\sup_t \mu^2_G(\alpha_j, \sigma_3)$	$\sup_t \mu_1(\alpha_j) \wedge \sup_t \mu^2_G(\alpha_j, \sigma_3)$
α_1	σ_3	(0.9, 0.1)	(0.9, 0.9)	(0.9, 0.1)*
α_2	σ_2	(0.8, 0.3)	(0.3, 0.8)	(0.3, 0.3)
α_3	σ_3	(0.5, 0.9)	(0.9, 0.9)	(0.5, 0.9)*
α_4	σ_1	(0.4, 0.6)	(0.8, 0.7)	(0.4, 0.6)
α_5	σ_2	(0.1, 0.1)	(0.3, 0.8)	(0.1, 0.1)
α_6	σ_1	(0.7, 0.3)	(0.8, 0.7)	(0.7, 0.3)*

* $\mu^1_G(\sigma_3)$

α_j	$f(\sigma_1, \alpha_j)$	$\sup_t \mu_0(\alpha_j)$	$\mu^1_G(\alpha_j, \sigma_1)$	$\sup_t \mu_1(\alpha_j) \wedge \mu^1_G(\alpha_j, \sigma_1)$
α_1	σ_1	(0.0, 0.1)	(0.8, 0.3)	(0.0, 0.1)
			(0.5, 0.8)	(0.0, 0.1)
α_2	σ_1	(0.3, 0.8)	(0.8, 0.3)	(0.3, 0.3)
			(0.5, 0.8)	(0.3, 0.8)*
α_3	σ_2	(0.7, 0.8)	(0.8, 0.3)	(0.7, 0.3)*
			(0.5, 0.7)	(0.5, 0.7)*
α_4	σ_2	(0.5, 0.5)	(0.8, 0.3)	(0.5, 0.3)
			(0.5, 0.7)	(0.5, 0.5)
α_5	σ_3	(0.3, 0.7)	(0.9, 0.1)	(0.3, 0.1)
			(0.5, 0.9)	(0.3, 0.7)
			(0.7, 0.3)	(0.3, 0.3)
α_6	σ_3	(0.2, 0.5)	(0.9, 0.1)	(0.2, 0.1)
			(0.5, 0.9)	(0.2, 0.5)
			(0.7, 0.3)	(0.2, 0.3)

* $\mu^0_G(\sigma_1)$

α_j	$f(\sigma_2, \alpha_j)$	$\sup_t \mu_0(\alpha_j)$	$\mu_G^1(\alpha_j, \sigma_2)$	$\sup_t \mu_1(\alpha_j) \wedge \mu_G^1(\alpha_j, \sigma_2)$
α_1	σ_2	(0.0, 0.1)	(0.8, 0.3)	(0.0, 0.1)
			(0.5, 0.7)	(0.0, 0.1)
α_2	σ_3	(0.3, 0.8)	(0.9, 0.1)	(0.3, 0.1)
			(0.5, 0.9)	(0.3, 0.8)
			(0.7, 0.3)	(0.3, 0.3)
α_3	σ_1	(0.7, 0.8)	(0.8, 0.3)	(0.7, 0.3)*
			(0.5, 0.8)	(0.5, 0.8)*
α_4	σ_3	(0.5, 0.5)	(0.9, 0.1)	(0.5, 0.1)
			(0.5, 0.9)	(0.5, 0.5)
			(0.7, 0.3)	(0.5, 0.3)
α_5	σ_1	(0.3, 0.7)	(0.8, 0.3)	(0.3, 0.3)
			(0.5, 0.8)	(0.3, 0.7)
α_6	σ_2	(0.2, 0.5)	(0.8, 0.3)	(0.2, 0.3)
			(0.5, 0.7)	(0.2, 0.5)

* $\mu_G^0(\sigma_2)$

α_j	$f(\sigma_3, \alpha_j)$	$\sup_t \mu_0(\alpha_j)$	$\mu_G^1(\alpha_j, \sigma_3)$	$\sup_t \mu_1(\alpha_j) \wedge \mu_G^1(\alpha_j, \sigma_3)$
α_1	σ_3	(0.0, 0.1)	(0.9, 0.1)	(0.0, 0.1)
			(0.5, 0.9)	(0.0, 0.1)
			(0.7, 0.3)	(0.0, 0.1)
α_2	σ_2	(0.3, 0.8)	(0.8, 0.3)	(0.3, 0.3)
			(0.5, 0.7)	(0.3, 0.7)
α_3	σ_3	(0.7, 0.8)	(0.9, 0.1)	(0.7, 0.1)
			(0.5, 0.9)	(0.5, 0.8)*
			(0.7, 0.3)	(0.7, 0.3)*
α_4	σ_1	(0.5, 0.5)	(0.8, 0.3)	(0.5, 0.3)
			(0.5, 0.8)	(0.5, 0.5)
α_5	σ_2	(0.3, 0.7)	(0.8, 0.3)	(0.3, 0.3)
			(0.5, 0.7)	(0.3, 0.7)
α_6	σ_1	(0.2, 0.5)	(0.8, 0.3)	(0.2, 0.3)
			(0.5, 0.8)	(0.2, 0.5)

* $\mu_G^0(\sigma_3)$

The result from the first iteration of the algorithm is as follows:

Initial State	$\mathcal{N}_1$	$[[\mathcal{N}_1]_{\mathcal{C}}]_2$
σ_1	$((\alpha_2,0),(\alpha_3,0))$	(0.3, 0.8)
	$((\alpha_3,0),(\alpha_2,6))$	(0.7, 0.3)
	$((\alpha_3,0),(\alpha_3,6))$	(0.5, 0.7)
σ_2	$((\alpha_3,0),(\alpha_2,7))$	(0.7, 0.3)
	$((\alpha_3,0),(\alpha_3,7))$	(0.5, 0.8)
σ_3	$((\alpha_3,0),(\alpha_3,0))$	(0.5, 0.8)
	$((\alpha_3,0),(\alpha_6,0))$	(0.7, 0.3)

Since for iteration one $\mathcal{A}_1 = \emptyset$, we get

Initial State	$\mathcal{T}_1 \equiv \mathcal{ND}\left[\left[\bigcup_{i=1}^{1}\mathcal{A}_i\right]_{\mathfrak{E}} \cup [\mathcal{N}_1]_{\mathcal{C}}\right]_1 = \mathcal{N}_1$	$\mathcal{C}(D)$	$\mathfrak{E}(D)$
σ_1	$((\alpha_2,0),(\alpha_3,2))$	(0.3, 0.8)	(0.3, 0.8)
	$((\alpha_3,0),(\alpha_2,6))$	(0.7, 0.3)	(0.7, 0.3)
	$((\alpha_3,0),(\alpha_3,6))$	(0.5, 0.7)	(0.5, 0.7)
σ_2	$((\alpha_3,0),(\alpha_2,7))$	(0.7, 0.3)	(0.7, 0.3)
	$((\alpha_3,0),(\alpha_3,7))$	(0.5, 0.8)	(0.5, 0.8)
σ_3	$((\alpha_3,0),(\alpha_3,0))$	(0.5, 0.8)	(0.5, 0.4)*
	$((\alpha_3,0),(\alpha_6,0))$	(0.7, 0.3)	(0.3, 0.3)*

* $\mathcal{A}_2 = \{((\alpha_3,0),(\alpha_3,0)),((\alpha_3,0),(\alpha_6,0))\}$

Since $\mathcal{A}_2$ is nonempty, we proceed to a second iteration (this time only for state σ_3).

α_j	$f(\sigma_3,\alpha_j)$	$\sup_t \mu_0(\alpha_j)$	$\mu_G^1(\alpha_j,\sigma_3)$	$\sup_t \mu_1(\alpha_j) \wedge \mu_G^1(\alpha_j,\sigma_3)$
α_1	σ_3	(0.0, 0.1)	(0.9, 0.1)	(0.0, 0.1)
			(0.5, 0.9)	(0.0, 0.1)
			(0.7, 0.3)	(0.0, 0.1)
α_2	σ_2	(0.3, 0.8)	(0.8, 0.3)	(0.3, 0.3)
			(0.5, 0.7)	(0.3, 0.7)*
α_3	σ_3	(0.7, 0.8)	(0.9, 0.1)	(0.7, 0.1)*
α_4	σ_1	(0.5, 0.5)	(0.8, 0.3)	(0.5, 0.3)
			(0.5, 0.8)	(0.5, 0.5)*
α_5	σ_2	(0.3, 0.7)	(0.8, 0.3)	(0.3, 0.3)
			(0.5, 0.7)	(0.3, 0.7)*
α_6	σ_1	(0.2, 0.5)	(0.8, 0.3)	(0.2, 0.3)
			(0.5, 0.8)	(0.2, 0.5)

* $\mu_G^0(\sigma_3)$

Initial State	$\mathcal{N}_2$	$[[\mathcal{N}_2]_{\mathcal{C}}]_2$
σ_3	$((\alpha_2,0),(\alpha_3,2))$	(0.3, 0.7)
	$((\alpha_3,0),(\alpha_1,0))$	(0.7, 0.1)
	$((\alpha_4,0),(\alpha_3,1))$	(0.5, 0.5)
	$((\alpha_5,0),(\alpha_3,5))$	(0.3, 0.7)

Since from iteration one we have:

Initial State	$\mathcal{A}_2$	$[[\mathcal{A}_2]_{\mathfrak{E}}]_2$
σ_3	$((\alpha_3,0),(\alpha_3,0))$	(0.5, 0.4)
	$((\alpha_3,0),(\alpha_6,0))$	(0.3, 0.3)

We get:

Initial State	$\mathcal{T}_2 \equiv \mathcal{ND}\left[\left[\bigcup_{i=1}^{2}\mathcal{A}_i\right]_{\mathfrak{E}} \cup [\mathcal{N}_2]_{\mathcal{C}}\right]_1$	$\mathcal{C}(D)$	$\mathfrak{E}(D)$
σ_3	$((\alpha_2,0),(\alpha_3,2))$	(0.3, 0.7)	(0.3, 0.7)
	$((\alpha_3,0),(\alpha_1,0))$	(0.7, 0.1)	(0.3, 0.1)*
	$((\alpha_4,0),(\alpha_3,1))$	(0.5, 0.5)	(0.5, 0.5)
	$((\alpha_5,0),(\alpha_3,5))$	(0.3, 0.7)	(0.3, 0.7)

* $\mathcal{A}_3 = \{((\alpha_3,0),(\alpha_1,0))\}$

Since $\mathcal{A}_3$ is nonempty, we proceed to a third iteration.

α_j	$f(\sigma_3,\alpha_j)$	$\sup_t \mu_0(\alpha_j)$	$\mu_G^1(\alpha_j,\sigma_3)$	$\sup_t \mu_1(\alpha_j) \wedge \mu_G^1(\alpha_j,\sigma_3)$
α_1	σ_3	(0.0, 0.1)	(0.9, 0.1)	(0.0, 0.1)
			(0.5, 0.9)	(0.0, 0.1)
			(0.7, 0.3)	(0.0, 0.1)
α_2	σ_2	(0.3, 0.8)	(0.8, 0.3)	(0.3, 0.3)
			(0.5, 0.7)	(0.3, 0.7)*
α_4	σ_1	(0.5, 0.5)	(0.8, 0.3)	(0.5, 0.3)
			(0.5, 0.8)	(0.5, 0.5)*
α_5	σ_2	(0.3, 0.7)	(0.8, 0.3)	(0.3, 0.3)
			(0.5, 0.7)	(0.3, 0.7)*
α_6	σ_1	(0.2, 0.5)	(0.8, 0.3)	(0.2, 0.3)
			(0.5, 0.8)	(0.2, 0.5)

* $\mu_G^0(\sigma_3)$

Initial State	$\mathcal{N}_3$	$[[\mathcal{N}_3]_{\mathcal{C}}]_2$
σ_3	$((\alpha_2,0),(\alpha_3,2))$	(0.3, 0.7)
	$((\alpha_4,0),(\alpha_3,1))$	(0.5, 0.5)
	$((\alpha_5,0),(\alpha_3,5))$	(0.3, 0.7)

Since from iteration two we have:

Initial State	$\mathcal{A}_3$	$[[\mathcal{A}_3]_{\mathfrak{E}}]_2$
σ_3	$((\alpha_3, 0), (\alpha_1, 0))$	(0.3, 0.1)

We get:

$\mathcal{T}_3 \equiv \mathcal{ND}\left[\left[\bigcup_{i=1}^{3} \mathcal{A}_i\right]_{\mathfrak{E}} \cup [\mathcal{N}_3]_{\mathcal{C}}\right]_1$	$\mathcal{C}(D)$	$\mathfrak{E}(D)$
$((\alpha_2, 0), (\alpha_3, 2))$	(0.3, 0.7)	(0.3, 0.7)
$((\alpha_4, 0), (\alpha_3, 1))$	(0.5, 0.5)	(0.5, 0.5)
$((\alpha_5, 0), (\alpha_3, 5))$	(0.3, 0.7)	(0.3, 0.7)

This time $\mathcal{A}_4$ is empty. Hence we're done! The solution to the MDP is:

Initial State	D	$\mathfrak{E}(D)$
σ_1	$((\alpha_2, 0), (\alpha_3, 0))$	(0.3, 0.8)
	$((\alpha_3, 0), (\alpha_2, 6))$	(0.7, 0.3)
	$((\alpha_3, 0), (\alpha_3, 6))$	(0.5, 0.7)
σ_2	$((\alpha_3, 0), (\alpha_2, 7))$	(0.7, 0.3)
	$((\alpha_3, 0), (\alpha_3, 7))$	(0.5, 0.8)
σ_3	$((\alpha_2, 0), (\alpha_3, 2))$	(0.3, 0.7)
	$((\alpha_4, 0), (\alpha_3, 1))$	(0.5, 0.5)
	$((\alpha_5, 0), (\alpha_3, 5))$	(0.3, 0.7)

Future research

This paper has established the theoretical basis for a dimension reduction algorithm for finding the non-dominated subset of a partially ordered set. An instance of one possible area of fertile application has also been given. There are however questions that naturally arise as a consequence of these results.

(1) The issue of the efficiency of the algorithm was raised in connection with the benefits that could accrue from a partitioning of the computational effort between a combinatorial step with a possibly simple cost structure and a direct evaluation and comparison step involving the more complex, "real" cost. Work needs to be done to make this claim more precise, both in terms of a theoretical analysis of computational complexity, and simulation studies of examples from a specific area of application.

(2) The choice of the bounding label $\mathcal{C}$ is crucial to the success of the algorithm in reducing computational cost. The example showed how this function could be chosen in one instance. It is of interest to investigate whether or not this is the best way of choosing this function in more general settings.

(3) Time-Dependent, Multi-Stage problems have been shown to be one possible area of fruitful application of this algorithm. As the main theoretical result has great generality (depending only on the existence of a labeling function, and a return space that is partially ordered), it is of interest to identify other classes of problems that may benefit from this approach.

References

[1] R.E. Bellman, On a Routing Problem. *Quarterly Appl. Math.* **16** (1958) 87-90.
[2] R.E. Bellman and L.A. Zadeh, Decision-Making in a Fuzzy Environment. *Management Sci.* **17** (1970) B-141-B-164.
[3] T.A. Brown and R.E. Strauch, Dynamic Programming in Multiplicative Lattices. *J. Math. Anal. Appl.* **12** (1965) 364-370.
[4] K.L. Cooke and E. Halsey, The Shortest Route through a Network with Time-Dependent Internodal Transit Times. *J. Math. Anal. Appl.* **14** (1966) 493-498.
[5] H.W. Corley and I.D. Moon, Shortest Paths in Networks with Vector Weights. *J. Opt. Theory Appl.* **46** (1985) 79-86.
[6] H.G. Daellenbach and C.A. DeKluyver, Note on Multiple Objective Dynamic Programming. *J. Oper. Res. Soc.* **31** (1980) 591-594.
[7] E.W. Djikstra, A Note on Two Problems in Connection with Graphs. *Numer. Math.* **1** (1959) 269-271.
[8] S.E. Dreyfus, An Appraisal of Some Shortest Path Algorithms. *Oper. Res.* **17** (1969) 395-412.
[9] S.E. ElMaghraby, The Concept of "State" in Discrete Dynamic Programming. *J. Math. Anal. Appl.* **29** (1970) 523-557.
[10] T. Getachew, M. Kostreva and L. Lancaster, A Generalization of Dynamic Programming for Pareto Optimization in Dynamic Network. *RAIRO: Oper. Res.* **34** (2000) 27-47.
[11] J. Halpern, Shortest Route with Time-Dependent Length of Edges and Limited Delay Possibilities in Nodes. *Z. Oper. Res.* **21** (1977) 117-124.
[12] M.I. Henig, The Principle of Optimality in Dynamic Programming with Returns in Partially Ordered Sets. *Math. Oper. Res.* **10** (1985) 462-470.
[13] 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 Technical Report 90-11 (1990).
[14] M.M. Kostreva and M.M. Wiecek, Time Dependency in Multiple Objective Dynamic Programming. *J Math. Anal. Appl.* **173** (1993) 289-308.
[15] A. Orda and R. Rom, Shortest-Path an Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length. *J. Assoc. Comp. Mach.* **37** (1990) 607-625.
[16] J. Kacprzyk and A.O. Esogbue, Fuzzy Dynamic Programming: Main Developments and Applications. *Fuzyy Sets Sys.* **81** (1996) 31-45.
[17] M.L. Hussein and M.A. Abo-Sinna, A Fuzzy Dynamic Approach to the Multicriterion Resource Allocation Problem. *Fuzzy Sets Sys.* **69** (1995) 115-124.