Open Journal of Discrete Mathematics, **2011, 1, 116-126**
doi:10.4236/ojdm.2011.13015 Published Online October 2011 (http://www.SciRP.org/journal/ojdm)

A Parametric Approach to the Bi-criteria Minimum Cost Dynamic Flow Problem

Mircea Parpalea
National College Andrei Şaguna, *Braşov*, *Romania*
E-mail: *parpalea@gmail.com*
Received June 26, 2011; *revised July* 28, 2011; *accepted August* 10, 2011

Abstract

This paper presents an algorithm for solving Bi-criteria Minimum Cost Dynamic Flow (BiCMCDF) problem with continuous flow variables. The approach is to transform a bi-criteria problem into a parametric one by building a single parametric linear cost out of the two initial cost functions. The algorithm consecutively finds efficient extreme points in the decision space by solving a series of minimum parametric cost flow problems with different objective functions. On each of the iterations, the flow is augmented along a cheapest path from the source node to the sink node in the time-space network avoiding the explicit time expansion of the network.

Keywords: Dynamic Network, Parametric Cost, Bi-Criteria Minimum Cost Flow, Successive Shortest Path

1. Introduction

Classical (static) network flow models have been well known as valuable tools for many applications [1] and therefore efficient algorithms have been developed. However, they fail to capture the dynamic property of many real-life problems, such as traffic planning, production and distribution systems, communication systems, and evacuation planning. Dynamic flows are widely used to model different network-structured, decision-making problems over time (see for example [2] and [3]), but because of their complexity, dynamic flow models have not been investigated as well as classical flow models. The time is an essential component, either because the flows take time to pass from one location to another, or because the structure of the network changes over time.

On the other hand, in many combinatorial optimization problems, the selection of the optimum solution takes into account more than one criterion. For example, in transportation problems or in network flows problems, the criteria that can be considered are the minimization of the cost for selected routes, the minimization of arrival time at the destinations, the minimization of the deterioration of goods, the minimization of the load capacity that would not be used in the selected vehicles, the maximization of safety, reliability, etc. Often, these criteria are in conflict and for this reason, a multi-objective network flow formulation of the problem is necessary.

In this paper, the case of bi-criteria minimum cost dynamic flow problem is considered. The proposed method consists in iteratively generating efficient extreme points in the decision space by solving a series of minimum parametric cost flow problems with different objective functions. On each of the iterations, the flow is augmented along a cheapest path from the source node to the sink node in the time-space network avoiding the explicit time expansion of the network.

Further on, in Section 2 some basic dynamic network flow terminology is presented together with some results used in the rest of the paper. More specialized terminology is developed in later sections. Section 3 deals with the bi-criteria minimum cost dynamic flow problem and with the parametric approach for solving it. In Section 4 the development of the proposed algorithm is presented while in Section 5, is given an example that helps understanding the steps performed by the former algorithm in a discrete dynamic network. In the presentation to follow, some familiarity with flow algorithms is assumed and many details are omitted, since they are straightforward modifications of known results.

2. Terminology and Notations

2.1. Dynamic Network Flows

Many dynamic network flow problems are considered as

extensions of static network flow problems. These include maximum dynamic flow and minimum cost dynamic flow problems. The maximum dynamic flow problem seeks a dynamic flow which sends as many as possible a commodity from a single source to a single sink of the network within the time horizon *T*. The minimum cost dynamic flow problem seeks a dynamic flow that minimizes the total shipment cost of a commodity in order to satisfy demands at certain nodes within *T*.

2.2. Discrete-Time Dynamic Network Flows

A discrete dynamic network $G=(N,A,T)$ is a directed graph where $N=\{\cdots,i,\cdots\}$ is a set of nodes i with $|N|=n$, $A=\{\cdots,a,\cdots\}$ is a set of arcs a with $|A|=m$, and T is a finite time horizon discretized into the set $H=\{0,1,\cdots,T\}$. An arc *a* from node *i* to node *j* is usually also denoted by (i,j). The following functions are associated with each arc $a=(i,j)\in A$: the time-dependent *capacity* (*upper bound*) function $u(i,j;\theta)$, $u: A\times\{0,1,\cdots,T\}\to\Re^+$ which represents the maximum amount of flow that can enter the arc (i,j) at time θ, the time-dependent *transit time* function $h(i,j;\theta)$, $h: A\times\{0,1,\cdots,T\}\to\aleph$, and the time-dependent *cost* function $c(i,j;\theta)$, $c: A\times\{0,1,\cdots,T\}\to\Re^+$ which represents the cost for sending one unit of flow through the arc (i,j) at time θ. Time is measured in discrete steps, so that if one unit of flow leaves node *i* at time θ on arc $a=(i,j)$, one unit of flow arrives at node *j* at time $\theta+h(i,j;\theta)$, where $h(i,j;\theta)$ is the transit time on arc *a*. The time horizon *T* is the time until which the flow can travel in the network. The *demand-supply* function $v(i;\theta)$, $v: N\times\{0,1,\cdots,T\}\to\Re$ represents the demand of node $i\in N$ at the time-moment $\theta\in\{0,1,\cdots,T\}$, if $v(i;\theta)<0$ or the supply of node *i* at the time-moment $\theta\in\{0,1,\cdots,T\}$, if $v(i;\theta)>0$. The network has two special nodes: a source node *s* with $v(s;\theta)\geq 0$ for $\theta\in\{0,1,\cdots,T\}$ and there exists at least one moment of time $\theta_0\in\{0,1,\cdots,T\}$ such that $v(s;\theta_0)>0$; and a sink node *t* with $v(t;\theta)\leq 0$ for $\theta\in\{0,1,\cdots,T\}$ and there exists at least one moment of time $\theta_1\in\{0,1,\cdots,T\}$ such that $v(t;\theta_1)<0$. The condition required for the flow to exist it that $\sum_{\theta\in\{0,1,\cdots,T\}}\sum_{i\in N} v(i;\theta)=0$

***Definition* 1**: [4] A *feasible dynamic flow* $f(i,j;\theta)$ *(feasible flow over time) on* $G=(N,A,u,h,c,T)$ *with time horizon T is a function* $f: A\times\{0,1,\cdots,T\}\to\Re^+$ *that satisfies the following constraints:*

$$\sum_{j|(i,j)\in A} f(i,j;\theta)-\sum_{\substack{j|(j,i)\in A\\ \theta-h(j,i;\theta)\geq 0}} f\left(j,i;\theta-h(j,i;\theta)\right)=v(i;\theta),$$

$$\forall i\in N;\ \forall\theta\in\{0,1,\cdots,T\} \tag{1.a}$$

$$0\leq f(i,j;\theta)\leq u(i,j;\theta),\ \forall\theta\in\{0,1,\cdots,T\},\ \forall(i,j)\in A; \tag{1.b}$$

$$f(i,j;\theta)=0,\ \forall(i,j)\in A,\ \theta\in\overline{T-h(i,j;\theta)+1,T} \tag{1.c}$$

where $f(i,j;\theta)$ *determines the rate of flow (per time unit) entering arc* (i,j) *at time* θ.

Capacity constraints (1.b) mean that in a feasible dynamic flow, at most $u(i,j;\theta)$ units of flow can enter the arc (i,j) at the time-moment θ. It is easy to observe that the flow does not enter arc (i,j) at time θ if it has to leave the arc after time *T*; this is ensured by condition (1.c). The total cost of the dynamic flow $f(i,j;\theta)$ in a discrete-time dynamic network is defined as:

$$C(f)=\sum_{\theta\in\{0,1,\cdots,T\}}\sum_{(i,j)\in A} f(i,j;\theta)\cdot c(i,j;\theta) \tag{2}$$

2.3. Time-Space Network

In the discrete time model, a useful tool for studying the minimum cost flow over time problem is the *time-space network*. The time-space network is a static network constructed by expanding the original network in the time dimension by considering a separate copy of every node $i\in N$ at every discrete time step in the time horizon *T*, $\theta\in\{0,1,\cdots,T\}$.

A *node-time pair* (NTP) (i,θ) refers to a particular node $i\in N$ at a particular time step $\theta\in\{0,1,\cdots,T\}$, *i.e.*, $(i,\theta)\in N\times\{0,1,\cdots,T\}$.

The NTP (i,θ_1) is said to be *linked* to the NTP (j,θ_2) if either

i) $(i,j)\in A$ and $\theta_2=\theta_1+h(i,j;\theta_1)$, or

ii) $(j,i)\in A$ and $\theta_1=\theta_2+h(j,i;\theta_2)$.

***Definition* 2**: [5] *The time-space network* G^T *of the original dynamic network G is defined as follows:*

$$N^T:=\left\{(i,\theta)\,|\,i\in N,\theta\in\{0,1,\cdots,T\}\right\}; \tag{3.a}$$

$$A^T:=\left\{a_\theta=\left((i,\theta),\left(j,\theta+h(i,j;\theta)\right)\right)\ \middle|\ 0\leq\theta\leq T-h(i,j;\theta)\right\},\ (i,j)\in A; \tag{3.b}$$

$$u^T(a_\theta):=u(a;\theta)\quad\text{for}\quad a_\theta\in A^T; \tag{3.c}$$

$$c^T(a_\theta):=c(a;\theta)\quad\text{for}\quad a_\theta\in A^T. \tag{3.d}$$

For every arc $(i,j)\in A$ with traversal time $h(i,j;\theta)$, capacity $u(i,j;\theta)$ and cost $c(i,j;\theta)$, the time-space network G^T contains arcs $\left((i,\theta),\left(j,\theta+h(i,j;\theta)\right)\right)$ for $\theta=0,1,\cdots,T-h(i,j;\theta)$ with capacities $u(i,j;\theta)$ and costs $c(i,j;\theta)$.

For the flow $f(a;\theta)$ in the dynamic network *G*, the function $f^T(a_\theta)$ that represents the corresponding flow in the time-space network G^T is defined as:

$$f^T(a_\theta) = f(a;\theta), \forall a_\theta \in A^T. \tag{4}$$

A *(discrete-time) dynamic path* is defined as a sequence of distinct, consecutively linked NTPs:

$$\begin{aligned} P(i_1,i_2):(i_1,\theta_1) &= (i_{k_1},\theta_{k_1}),(i_{k_2},\theta_{k_2}),\cdots,(i_{k_q},\theta_{k_q}) \\ &= (i_2,\theta_2). \end{aligned} \tag{5}$$

2.4. Time-Dependent Residual Network

The *time-dependent residual network* corresponding to a feasible flow *f* can be viewed as the static residual network of the time-space network corresponding to the dynamic network.

For $f(i,j;\theta)$ being the flow entering arc (i,j) at time θ, an additional flow $u(i,j;\theta)-f(i,j;\theta)$ departing from node *i* at time θ to node *j* along the arc (i,j) can be sent. Also, $f(i,j;\theta)$ units of flow can be sent from node *j* departing at time $\theta+h(i,j;\theta)$ and consequently arriving at node *i* at time θ over the arc (i,j), which amounts to cancelling the existing flow on the arc. Here, an arc with negative travel time (*i.e.* departing at $\theta+h(i,j;\theta)$ and arriving at θ) is considered. Whereas sending a unit of flow from *i* at time θ to *j* along (i,j) increases the flow cost by $c(i,j;\theta)$ units, sending a unit of flow in reverse direction from *j* departing at time $\theta+h(i,j;\theta)$ to *i* on the same arc decreases the flow cost by $c(i,j;\theta)$ units.

Considering the above mentioned ideas, the residual network with respect to a current dynamic flow *f* is defined as follows.

***Definition* 3**: [5] *The residual dynamic network with respect to a given feasible dynamic flow f is defined as* $G(f) := (N, A(f), T)$ with $A(f) := A^+(f) \cup A^-(f)$ *where*

$$A^+(f) := \{(i,j) \mid (i,j) \in A,\ \exists\, \theta \le T - h(i,j;\theta)\} \quad \text{with} \quad u(i,j;\theta) - f(i,j;\theta) > 0 \tag{6.a}$$

$$A^-(f) := \{(i,j) \mid (j,i) \in A,\ \exists\, \theta \le T - h(j,i;\theta)\} \quad \text{with} \quad f(j,i;\theta) > 0 \tag{6.b}$$

While the direct arcs $(i,j) \in A^+(f)$ have the same transit times $h(i,j;\theta)$ and costs $c(i,j;\theta)$ as in the original dynamic network *G*, the artificial reverse arcs $(i,j) \in A^-(f)$ in the residual dynamic network $G(f)$ are provided with the following attributes:

$$h(i,j;\theta + h(j,i;\theta)) := -h(j,i;\theta), \tag{7}$$

$$c(i,j;\theta + h(j,i;\theta)) := -c(j,i;\theta), \tag{8}$$

with $(j,i) \in A, 0 \le \theta + h(j,i;\theta) \le T, f(j,i;\theta) > 0$.

The residual capacities of the arcs (i,j) in the residual dynamic network $G(f)$ are defined as follows:

$$r(i,j;\theta) := u(i,j;\theta) - f(i,j;\theta), \quad (i,j) \in A, 0 \le \theta + h(i,j;\theta) \le T \tag{9.a}$$

$$r(i,j;\theta + h(j,i;\theta)) = f(j,i;\theta), \quad (j,i) \in A, 0 \le \theta + h(j,i;\theta) \le T \tag{9.b}$$

***Definition* 4**: [6] *A dynamic path*

$$P(s = i_1, i_2, \cdots, i_q = i)$$

from node s to node i is said to be a dynamic augmenting path *if* $r(i_k, i_{k+1}; \theta_k) > 0$ *for* $(i_k, i_{k+1}) \in A(f)$ *and* $k = 1, \cdots, q-1$.

***Definition* 5**: [6] *Given a dynamic flow f, the* residual capacity *of a dynamic augmenting path*

$$P(s = i_1, i_2, \cdots, i_q = i)$$

is defined by:

$$r(P) := \min_{1 \le k \le q-1} r(i_k, i_{k+1}; \theta_k), \tag{10}$$

for $(i_k, i_{k+1}) \in A(f)$, $k = 1, \cdots, q-1$.

***Definition* 6**: [6] *The* cost *of a dynamic augmenting path* $P(s = i_1, i_2, \cdots, i_q = i)$ *is defined by*:

$$C(P) := \sum_{(i_k, i_{k+1}) \in A(f)} c(i_k, i_{k+1}; \theta_k) \quad \text{for} \quad k = 1, \cdots, q-1$$

A dynamic augmenting path $P(s = i_1, i_2, \cdots, i_q = i)$ is referred to as a *dynamic shortest augmenting path* (DSAP) from node $s = i_1$ to node $i_q = i$ if $C(P) \le C(P')$ for all dynamic augmenting paths P' from node *s* to node *i*.

A dynamic path $P(i_1, i_q) : (i_1, \theta_1), (i_2, \theta_2), \cdots, (i_q, \theta_q)$ is called a *dynamic cycle* if $i_q = i_1$ and $\theta_q = \theta_1$. A *negative cycle* is defined as a dynamic cycle whose total cost is negative and whose capacity is greater than zero.

3. Bi-Criteria Minimum Cost Dynamic Flow Problem

The bi-criteria minimum cost dynamic flow problem is to determine how a given amount of flow that simultaneously minimizes two total costs should be sent from a source node to a sink node within the time horizon *T*, subject to the capacity limits on the arcs of the network. The successive shortest path approach adapted to the dynamic residual network is based on solving a series of successive shortest path problems, where each is solved in a residual time-space network. An amount of flow equal to the capacity of each minimum cost path obtained is augmented, until the entire flow has been sent

from the source to the sink. The main difference among the algorithms consists in solving the shortest path problem in the dynamic residual network.

3.1. The Problem Formulation

Let $G=(N,A,T)$ be a dynamic network with a node set *N*, an arc set *A* and a finite time horizon T discretized into the set $\{0,1,\cdots,T\}$. Without loss of generality, we consider that no arcs are entering in the source node or leaving the sink node. Considering the time-dependent *capacity* (*upper bound*) function $u(i,j;\theta)$, $u: A\times\{0,1,\cdots,T\}\to\Re^+$, the time-dependent *transit time* function $h(i,j;\theta)$, $h: A\times\{0,1,\cdots,T\}\to\aleph$, and the time-dependent *cost* functions $c_k(i,j;\theta)$, $c_k: A\times\{0,1,\cdots,T\}\to\Re^+$, representing the cost associated with the *k-th* objective function, $k=1,2$, for sending one unit of flow through the arc (i,j) at time θ, the BiCMCDF problem can be stated as finding the flow function $f: A\times\{0,1,\cdots,T\}\to\Re^+$ that satisfies the following constraints:

$$\text{minimize } y_k(f)=\sum_{\theta=0}^{T}\sum_{(i,j)\in A}c_k(i,j;\theta)\cdot f(i,j;\theta), \quad k=1,2; \tag{11.a}$$

subject to:

$$\sum_{\theta=0}^{T}\sum_{(i,t)\in A}\sum_{\vartheta\,|\,\vartheta+h(i,t;\vartheta)=\theta} f(i,t;\vartheta)=v \tag{11.b}$$

$$\sum_{j\,|\,(i,j)\in A} f(i,j;\theta)-\sum_{j\,|\,(j,t)\in A}\sum_{\vartheta\,|\,\vartheta+h(j,i;\vartheta)=\theta} f(j,i;\vartheta)=0, \quad \forall i\in N-\{s,t\} \tag{11.c}$$

$$0\le f(i,j;\theta)\le u(i,j;\theta),\ \forall\theta\in\{0,1,\cdots,T\},\ \forall(i,j)\in A. \tag{11.d}$$

The value of the dynamic flow for a time horizon *T* is denoted by *v*. Any vector *f* that satisfies the constraint (11.b), the flow conservation constraint (11.c) at the different node-time pairs and the bound constraint (11.d) is called a *feasible solution* of the bi-criteria minimum cost dynamic flow (BiCMCDF) problem.

The set of feasible solutions or *decision space* is denoted by *F* and its image through

$$Y(F)=\{(y_1(f),y_2(f))\,|\,f\in F\}$$

is called *objective space*.

In general, there is no feasible solution of the (BiCMCDF) problem that simultaneously minimizes both objectives. In other words, an optimum global solution does not exist. For this reason, the solutions of these problems are searched for among the set of efficient points.

***Definition* 7**: [7] *A feasible solution* $f\in F$ *of the bi-criteria minimum cost flow problem is called* efficient *if, and only if, there does not exist another feasible solution* $f'\in F$ *so that* $y_k(f')\le y_k(f)$ *for all k values and* $y_k(f')<y_k(f)$ *for at least one k.*

***Definition* 8**: [7] $Y(f)$ *is a* non-dominated *criterion vector if* f *is an efficient solution. Otherwise* $Y(f)$ *is a dominated criterion vector*.

The set of efficient solutions of *F* will be denoted by $E[F]$ while, by extension, $E[Y(F)]$ is called the set of non-dominated solutions of $Y(F)$. It is well known that to characterize $E[Y(F)]$ for the bi-criteria continuous minimum cost flow problem, it is only necessary to identify the extreme efficient points of $Y(F)$. The set of efficient extreme points of *F* will be denoted by and by $E_{ex}[F]$ and the corresponding points of $Y(F)$ will be denoted by $E_{ex}[Y(F)]$.

3.2. The Parametric Approach

For the bi-criteria linear programming (BCLP) problems, Gass and Saaty [8] provide an algorithm using the parametric programming technique. Geoffrion [9] discusses the availability of parametric programming for a broader class of bi-criteria problems. The functions $y_1(f)$ and $y_2(f)$ are assumed to be convex and the feasible region *F* is a compact convex set. The parametric programming problem is defined as:

$$\begin{aligned}&\text{minimize}\quad y(f)=\lambda\cdot y_1(f)+(1-\lambda)\cdot y_2(f)\\&\text{subject to}\quad f\in F,\quad \text{for}\ \ 0\le\lambda\le 1\end{aligned} \tag{12}$$

He's procedure is not radically different from that of Gass and Saaty [8].

***Lemma* 1**: [9] *If* f_0 *is efficient, then there exists a scalar* λ_0 *in the unit interval such that* f_0 *is an optimal solution of the parametric programming problem*.

***Theorem* 1**: [9] *The set of all efficient extreme points of the bi-criteria minimum cost flow problem can be found by solving* (12) *for each* λ *in the unit interval.*

***Lemma* 2**: [9] *For each fixed value of* λ *satisfying* $0<\lambda<1$, *the optimal solution of* (12) *is a compact line segment in the objective space. If* $y(f_1)$ *and* $y(f_2)$ *are the endpoints of the line segment, then*

$$y(t\cdot f_1+(1-t)\cdot f_2)=t\cdot y(f_1)+(1-t)\cdot y(f_2),$$

for all $t,\ \ 0\le t\le 1$.

Aneja and Nair [10] developed a simple algorithm for bi-criteria transportation problems. Their procedure generates efficient extreme points on the objective space $Y(F)$ rather than on the decision space. A series of single objective problems are solved with different objective functions and each problems leads to either a new

efficient extreme point or changes the direction of search in the objective space. Although the procedure is conceptually simple, it doesn't provide the λ-regions for each efficient point. Lee and Pulat [7] used the parametric programming procedure for the bi-criteria minimum cost flow problem by modifying the out-of-kilter algorithm.

The approach that was proposed in this paper finds the efficient points in the decision space using a successive dynamic shortest augmenting path algorithm based on a linear parametric label setting procedure.

Instead of the two costs functions $c_1(i,j;\theta)$ and $c_2(i,j;\theta)$, a single parametric cost function of λ, $c(i,j;\theta;\lambda)$ can be defined as follows:

$$c(i,j;\theta;\lambda) = (1-\lambda)\cdot c_1(i,j;\theta) + \lambda \cdot c_2(i,j;\theta), \quad \lambda \in [0,1] \tag{13}$$

As it can easily be seen, for $\lambda_0 = 0$ the value of the parametric cost equals the cost associated with the first objective function, while for $\lambda = 1$ the cost associated with the second objective function is obtained.

The algorithm starts with $\lambda_0 = 0$ and, in any of its labeling steps, lexicographically finds the minimum cost associated with the first objective function and the minimum cost associated with the second objective function, *i.e.* $\min(c_1, c_2)$.

In the more general case, relating to a given value λ_k of the parameter, the parametric linear cost function $c(i,j;\theta;\lambda)$ can be rewritten as:

$$c(i,j;\theta;\lambda) = \alpha_k(i,j;\theta) + (\lambda - \lambda_k)\cdot \beta(i,j;\theta), \quad \lambda \in [\lambda_k, 1] \tag{14}$$

with $\alpha_k(i,j;\theta) = c_1(i,j;\theta) + \lambda_k \cdot (c_2(i,j;\theta) - c_1(i,j;\theta))$ being the value of the parametric cost function $c(i,j;\theta;\lambda)$ for $\lambda = \lambda_k$ and

$$\beta(i,j;\theta) = c_2(i,j;\theta) - c_1(i,j;\theta)$$

being the slope of the parametric cost function for $\lambda \geq \lambda_k$.

Similarly, the parametric cost function of a dynamic augmenting path $P(q)$ from the start node *s* to node *q* can be written as:

$$c(P(q),\lambda) = \pi_k(q,\theta) + (\lambda - \lambda_k)\cdot \tau(q,\theta), \quad \lambda \in [\lambda_k, 1] \tag{15}$$

with $\pi_k(q,\theta) = \sum_{(i,j)\in P(q)} \alpha_k(i,j;\theta)$ being the value of the parametric cost function $c(P(q),\lambda)$ for $\lambda = \lambda_k$ and $\tau(q,\theta) = \sum_{(i,j)\in P(q)} \beta(i,j;\theta)$ being the slope of the parametric cost function for $\lambda_k \leq \lambda < \lambda_{k+1}$.

Moreover, in every labeling step, the value λ_{k+1} of the parameter by which one of the two linear parametric cost functions of λ which are compared remains minimum can be computed as:

$$\lambda_{k+1} = \lambda_k + (\pi_k(j,\theta) - \pi_k(i,\theta)) / (\tau(i,\theta) - \tau(j,\theta)) \tag{16}$$

***Lemma* 3**: *The set of all efficient extreme points of the bi-criteria minimum cost flow problem can be found by solving a classical minimum cost flow problem with the parametric cost function*

$$c(i,j;\theta;\lambda) = (1-\lambda)\cdot c_1(i,j;\theta) + \lambda \cdot c_2(i,j;\theta)$$

for successive λ_k *values in the unit interval.*

Proof: The proof of the lemma results directly from theorem 1 by simply making the replacement $t := 1-\lambda$ in lemma 2. □

4. Development of the Algorithm

4.1. Parametric Shortest Dynamic Paths

Solution approaches for bi-criteria shortest path problems are divided into two classes: label-setting and label-correcting [11]. Label setting algorithms can only be applied on acyclic networks and networks with nonnegative. The time-dependent residual network is composed of two sub-networks: a forward network consisting of the set of forward arcs, denoted by $A^+(f)$, having positive travel times and travel costs; and a reverse network consisting of the set of reverse arcs, denoted by $A^-(f)$ and having negative travel times and travel costs. Each of the two sub-networks, alone, is acyclic. In exploring the time-dependent residual network, the forward and reverse arcs are explored simultaneously.

***Property* 1**: [12] *If a time-space network contains no negative cycle, then the time-dependent residual network generated based on a dynamic shortest augmenting path contains no negative cycle.*

The basic idea of the label setting algorithm is to start from NTP (s,θ_s) and to label the NTPs which are reachable from (s,θ_s), according to their cost from (s,θ_s). The algorithm maintains a parametric cost label with each NTP (i,θ) memorized in the set $\Pi(i,\theta) := \{\pi(i,\theta), \tau(i,\theta)\}$. For every NTP (i,θ), the distance label $\pi(i,\theta)$ is either ∞, indicating that it was not yet discovered any augmenting path from (s,θ_s) to (i,θ), or it is the length (cost) of the shortest augmenting path to (i,θ).

At any point in the algorithm, the distance labels are divided in two groups: *permanent* and *temporary*. The label $\pi(i,\theta)$ is permanent once it denotes the length of shortest augmenting path from (s,θ_s) to (i,θ), otherwise it is temporary. A set *L* of candidate nodes with

temporary labels is maintained, which initially includes only the source node. The set *L* holds, in increasing order of their temporary labels, all node-time pairs which have been reached so far by the algorithm and which are to be visited. At any iteration, the algorithm selects a node-time pair (i,θ) with minimum temporary label, makes its distance label permanent, checks optimality conditions and updates labels accordingly.

Optimality Conditions

For a given value λ_k of the parameter, the distance labels $\pi(i,\theta)$ represent the length of shortest augmenting paths from NTP (s,θ_s) to NTP (j,θ) if they satisfy the following:

a) $\forall (i,j) \in A,\ f(i,j;\vartheta) < u(i,j;\vartheta)$ and $\theta = \vartheta + h(i,j;\vartheta)$ if either
   i) $\pi(j,\theta) < \pi(i,\vartheta) + \alpha_k(i,j;\vartheta)$, or
   ii) $\pi(j,\theta) = \pi(i,\vartheta) + \alpha_k(i,j;\vartheta)$ and $\tau(j,\theta) \le \tau(i,\vartheta) + \beta(i,j;\vartheta)$

b) $\forall (j,i) \in A,\ f(j,i;\theta) > 0$ if either
   i) $\pi(j,\theta) < \pi(i,\theta + h(j,i;\theta)) - \alpha_k(j,i;\theta)$, or
   ii) $\pi(j,\theta) = \pi(i,\theta + h(j,i;\theta)) - \alpha_k(j,i;\theta)$ and $\tau(j,\theta) \le \tau(i,\theta + h(j,i;\theta)) - \beta(j,i;\theta)$

The minimum cost labels $\pi(i,\theta)$ of all node-time pairs are initialised to infinity with the exception of the minimum cost labels of the source node which are initialised to zero, $\pi(s,\theta) := 0$, $\forall\, \theta \in \{0,1,\cdots,T\}$. For every node-time pair (i,θ) selected from *L*, the arcs with positive residual capacity connecting (i,θ) to (j,ϑ) are explored, where $0 < \vartheta = \theta + h(i,j;\theta) \le T$ if the arc connecting (i,θ) to (j,ϑ) is a forward arc and $0 \le \vartheta = \theta - h(j,i;\vartheta) \le T$ if it is a reverse arc. Then the minimum cost labels are updated and the node-time pair (j,ϑ) is added to the candidate set if it is not already in *L*. The process is repeated until there are no more candidate nodes in *L*.

The travel cost of the minimum cost path, computed based on predecessor vector *p*, is given by

$$\overline{\pi}(t) = \min_{\theta \in \{0,1,\cdots,T\}} \{\pi(t,\theta)\}$$

with

$$\overline{\tau}(t) = \min_{\theta \in \{0,1,\cdots,T\}} \{\tau(t,\theta) \mid \pi(t,\theta) = \overline{\pi}(t)\}.$$

The Parametric Shortest Dynamic Path (PSDP) procedure is presented in **Table 1**.

Procedure **next_lambda** $(\pi_1(i), \pi_2(i), \tau_1(i), \tau_2(i))$ presented in **Table 2**, returns the value of the parameter up to which one of the two linear parametric cost functions of λ remains minimum. The two linear parametric functions to be compared, regarding the arguments of the function, are $\pi_1(i) + (\lambda - \lambda_k) \cdot \tau_1(i)$ and

$$\pi_2(i) + (\lambda - \lambda_k) \cdot \tau_2(i).$$

***Theorem* 2**: *The complexity of Dynamic Parametric Shortest Path (DPSP) procedure is* $O(nmT^2)$.

Proof: The algorithm performs $O(nT)$ iterations (selections) and in each of the iterations $O(mT)$ arcs are explored (which corresponds to the number of arcs in the time-space network). Hence, the total complexity of the (DPSP) procedure is $O(nmT^2)$. □

4.2. Successive Parametric Shortest Path Algorithm

Each step of the successive shortest path algorithm for the bi-criteria minimum cost dynamic flow problem will repeatedly perform the following operations:

i) Compute a parametric shortest dynamic path *P* from the source node to the sink node;

ii) Find the residual capacity $r(P)$ of the minimum cost path;

iii) Augment the flow along the parametric shortest dynamic path and update the residual network.

For a given value λ_k of the parameter, the algorithm computes the values of the parametric costs $\alpha_k(i,j;\theta) = c_1(i,j;\theta) + \lambda_k \cdot (c_2(i,j;\theta) - c_1(i,j;\theta))$ for $\lambda = \lambda_k$ and the slopes of the parametric cost functions

$$\beta(i,j;\theta) = c_2(i,j;\theta) - c_1(i,j;\theta)$$

for all arcs in the time-dependent residual network. Then the algorithm successively finds parametric shortest dynamic paths and increases the flow until the value of the dynamic flow for the time horizon *T* equals the total deficit of all sink node-time pairs, υ. In each of the iterations, the value of the parameter by which the parametric shortest dynamic path remains minimal is computed and then the algorithm reiterates with this new value of the parameter. The algorithm will terminate when the value of the parameter becomes equal to 1. The Successive Parametric Shortest Path (SPSP) algorithm is presented in **Table 3**.

***Theorem* 3**: *The Successive Parametric Shortest Path* (*SPSP*) *algorithm computes correctly a bi-criteria minimum cost dynamic flow for a given time horizon T.*

Proof: The consecutive λ_k values are computed as the closest values of the parameter for which the order of the parametric linear cost functions do not reverse, i.e. do not have crossing points within the interval $(\lambda_k, \lambda_{k+1})$. Since for a λ_k value, the flow is augmented along successive shortest paths, the correctness of the algorithm results from the classical (non-parametric) algorithm. For consecutive λ_k values of the parameter, the proof results directly from Lemma 3. □

A series of single objective problems are solved with different objective functions, corresponding to different

Table 1. Dynamic Parametric Shortest Path (DPSP) procedure.

```
(1)  procedure PSDP(p, λ_k, λ_{k+1}, B);
(2)  begin
(3)     for all θ ∈ {0,1,...,T} do
(4)       begin
(5)          π(s,θ) := 0;   τ(s,θ) := 0;
(6)          for all i ∈ N−{s} do
(7)             begin
(8)                π(i,θ) := ∞;  τ(i,θ) := ∞;
(9)             end;
(10)      end;
(11)    π̄(t) := ∞;  τ̄(t) := ∞;  L := {(s,θ) | θ = 0,1,...,T};
(12)    while (L ≠ φ) do
(13)      begin
(14)        select (i,θ_i) with minimum π(i,θ_i) from L;   L = L − {(i,θ_i)};
(15)        for all j ∈ A⁺(i) with r(i,j;θ_i) > 0 do
(16)          if (θ_i + h(i,j;θ_i) ≤ T) then
(17)            begin
(18)              θ_j := θ_i + h(i,j;θ_i);
(19)              λ_{k+1} := next_lambda((π(j,θ_j), π(i,θ_i)+α(i,j;θ_i), τ(j,θ_j), τ(i,θ_i)+β(i,j;θ_i), λ_{k+1});
(20)              if ((π(i,θ_i) + α(i,j;θ_i) < π(j,θ_j))) or
                  ((π(i,θ_i) + α(i,j;θ_i) = π(j,θ_j)) and (τ(i,θ_i) + β(i,j;θ_i) < τ(j,θ_j))) then
(21)                begin
(22)                  π(j,θ_j) := π(i,θ_i) + α(i,j;θ_i);  τ(j,θ_j) := τ(i,θ_i) + β(i,j;θ_i);
(23)                  p(j,θ_j) := (i,θ_i);
(24)                  if ((j,θ_j) ∉ L) then L := L ∪ {(j,θ_j)};
(25)                end;
(26)            end;
(27)        for all j ∈ A⁻(i) do
(28)        for all θ_j such that θ_i = θ_j + h(j,i;θ_j) and r(j,i;θ_j) < u(j,i;θ_j) do
(29)          begin
(30)            λ_{k+1} := next_lambda((π(j,θ_j), π(i,θ_i)−α(j,i;θ_j), τ(j,θ_j), τ(i,θ_i)−β(j,i;θ_j), λ_{k+1});
(31)            if ((π(i,θ_i) − α(j,i;θ_j) < π(j,θ_j)) or
                ((π(i,θ_i) − α(j,i;θ_j) = π(j,θ_j)) and (τ(i,θ_i) − β(j,i;θ_j) < τ(j,θ_j))) then
(32)              begin
(33)                π(j,θ_j) := π(i,θ_i) − α(j,i;θ_j);  τ(j,θ_j) := τ(i,θ_i) − β(j,i;θ_j);
(34)                p(j,θ_j) := (i,θ_i);
(35)                if ((j,θ_j) ∉ L) then L := L ∪ {(j,θ_j)};
(36)              end;
(37)          end;
(38)      end;
(39)    π̄(t) = min_{θ∈{0,1,..,T}} {π(t,θ)};   τ̄(t) = min_{θ∈{0,1,..,T}} {τ(t,θ) | π(t,θ) = π̄(t)};
(40)    if (π̄(t) = ∞) then B := 0
(41)    else for all θ ∈ {0,1,...,T} do
(42)      λ_{k+1} := next_lambda(π̄(t), π(t,θ), τ̄(t), τ(t,θ), λ_{k+1});
(43) end;
```


Table 2. Procedure next_lambda.

```
(1)  procedure next_lambda(π1(i),π2(i),τ1(i),τ2(i),λk+1);
(2)  begin
(3)      λ":= λk+1;
(4)      if (τ2(i)-τ1(i))≠0 then
(5)        begin
(6)          λ':= λk+(π1(i)-π2(i))/(τ2(i)-τ1(i));
(7)          if ((λ'>λk) and (λ'<λk+1)) then λ":=λ';
(8)        end;
(9)      return(λ");
(10) end;
```

Table 3. Successive Parametric Shortest Path (SPSP) algorithm.

```
(1)  SPSP(G,v);
(2)  begin
(3)     k:=0; λk:=0;
(4)     for all θ∈{0,1,...,T} do
(5)        for all (i,j)∈A do β(i,j;θ):=c2(i,j;θ)-c1(i,j;θ);
(6)     while (λk<1) do
(7)       begin
(8)         λk+1:=1; v':=v;
(9)         for all θ∈{0,1,...,T} do
(10)        for all (i,j)∈A do
(11)          begin
(12)            fk(i,j;θ):=0; αk(i,j;θ)=c1(i,j;θ)+λk·(c2(i,j;θ)-c1(i,j;θ));
(13)          end;
(14)        while (v'>0) do
(15)          begin
(16)             B:=1;
(17)             for all θ∈{0,1,...,T} do
(18)               begin
(19)                 p(s,θ):=0;
(20)                 for all i∈N-{s} do p(i,θ):=-1;
(21)               end;
(22)             PSDP(p,λk,λk+1,B);
(23)             if (B=0)then STOP (no path can be found)
(24)             else
(25)               begin
(26)                 build path P based on p;
(27)                 r(P):=min{r(i,j;θi)}; δ:=min{v',r(P)}; v':=v'-δ;
                          (i,j)∈P
(28)                 for all arcs (i,j)∈P do
(29)                   begin
(30)                     if (θj>θi) then r(i,j;θi):=r(i,j;θi)-δ
(31)                                else r(j,i;θj):=r(j,i;θj)+δ;
(32)                     compute fk(i,j;θi);
(33)                   end;
(34)               end;
(35)          end;
(36)        Yk:={(y1(fk),y2(fk))};
(37)        k:=k+1;
(38)      end;
(39) end;
```


λ_k values, and each problem leads (for the non-degenerate case) to a new efficient solution. The algorithm terminates when the value of the parameter reaches to the maximum value of the unit interval. Starting with $\lambda_k = 0$ for $k = 0$ and ending when $\lambda_k = 1$ for $k = K$, the algorithm performs K steps corresponding to the K linear segments between two consecutive efficient extreme points.

***Theorem* 4**: *The complexity of the Successive Parametric Shortest Path* (*SPSP*) *algorithm for computing the set of efficient extreme points in the decision space is* $O\left(K \cdot nmT^2 v\right)$.

Proof: For the labelling operation and computing parametric costs, all arcs at all times have to be examined, so the running time is $O(mT)$. Updating the residual networks after augmenting the flow also requires a running time of $O(mT)$. Since at most υ augmentations are done by the algorithm, procedure DPSP is called υ times for every λ_k value of the parameter. Hence an efficient extreme point is computed in $O(mT)$ + $O(mT) + \upsilon \cdot O\left(nmT^2\right)$, *i.e.* $\upsilon \cdot O\left(nmT^2\right)$. Consequently, for the *K* steps corresponding to the *K* linear segments between two consecutive efficient extreme points, the total complexity of the SPSP algorithm is $O\left(K \cdot nmT^2 \upsilon\right)$. □

5. Example

In the discrete-time dynamic network presented in **Figure 1(a)**, the problem is to send $\upsilon = 3$ units of flow at minimum bi-criteria cost from the source node *s* (node 1) to the sink node *t* (node 5) within a time horizon which is set to $T = 4$. The upper bounds (capacity) of all arcs are set to $u(i,j;\theta) := 2$, $\forall (i,j) \in A$, $\forall \theta \in \{0,1,2,3,4\}$ and the transit times and costs are presented in **Table 4**.

In the initialisation step, the slope $\beta(i,j;\theta) = c_2(i,j;\theta) - c_1(i,j;\theta)$ of the parametric costs functions are computed (as presented in **Table 4**) and, for $k = 0$ and the corresponding initial value of the parameter $\lambda_0 = 0$, $\alpha_0(i,j;\theta) := c_1(i,j;\theta)$ are set. The predecessor vector is initialised at all time values to $p(i,\theta) := -1$ for all nodes except for the source node, for which $p(s,\theta)$ is set to zero, and procedure **PSDP** is called.

***Iteration* 1**: The distance labels are initialised to $\pi(1,\theta) := 0$ for $\forall \theta \in \{0,1,\cdots,T\}$ and the set *L* of candidate nodes is set to $L := \{(1,0),(1,1),(1,2),(1,3),(1,4)\}$. The selected node-time pair, (1,0) is removed from the

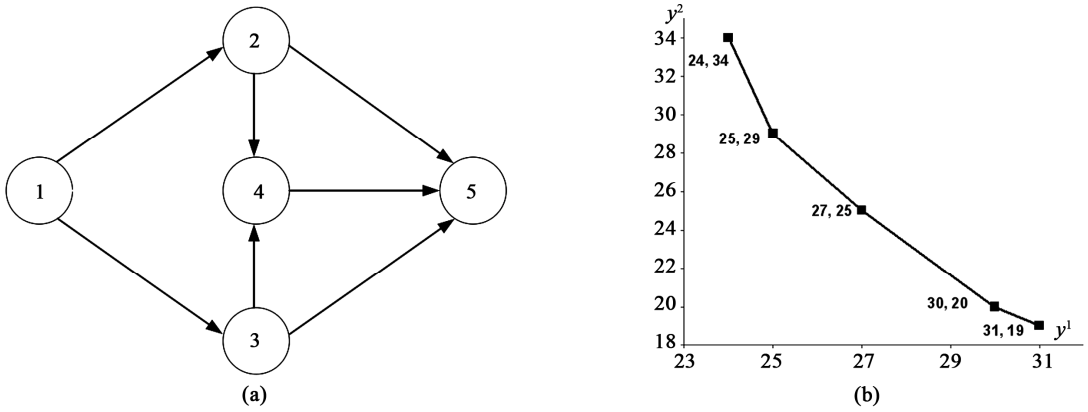

Figure 1. (a) The dynamic network *G* considered for exemplifying how the Successive Parametric Shortest Path (SPSP) algorithm works; (b) The set of all non-dominated points which lie on the efficient boundary in the objective space for the bi-criteria minimum cost dynamic flow problem in network *G* .

Table 4. Transit times and costs on arcs for the dynamic network in Figure 1.

(*i*,*j*)	(1,2)	(1,3)	(2,4)	(2,5)	(3,4)	(3,5)	(4,5)
$h(i,j;\theta)$	$\begin{cases}2, & 0 \le \theta < 1\\ 3, & \theta \ge 1\end{cases}$	$\begin{cases}1, & 0 \le \theta < 2\\ 2, & \theta \ge 2\end{cases}$	$\begin{cases}3, & 0 \le \theta < 2\\ 1, & \theta \ge 2\end{cases}$	1	$\begin{cases}2, & 0 \le \theta < 2\\ 1, & \theta \ge 2\end{cases}$	$\begin{cases}1, & 0 \le \theta < 2\\ 3, & \theta \ge 2\end{cases}$	1
$c_1(i,j;\theta)$	2	2	7	9	$\begin{cases}4, & 0 \le \theta < 2\\ 5, & \theta \ge 2\end{cases}$	7	1
$c_2(i,j;\theta)$	3	4	2	2	$\begin{cases}6, & 0 \le \theta < 2\\ 1, & \theta \ge 2\end{cases}$	5	5
$\beta(i,j;\theta)$	1	2	–5	–7	$\begin{cases}2, & 0 \le \theta < 2\\ -4, & \theta \ge 2\end{cases}$	–2	4

Table 5. The steps performed by Successive Parametric Shortest Path (SPSP) algorithm on the dynamic network in Figure 1.

k	λ_k	P	δ	Y_k	λ_{k+1}
0	0	$P:=((1,0),(3,1),(4,3),(5,4))$	2	$Y_0=(24,34)$	$\lambda_1=1/6$
		$P:=((1,1),(3,2),(4,3),(3,1),(5,2))$	1		
1	1/6	$P:=((1,1),(3,2),(4,3),(5,4))$	2	$Y_1=(25,29)$	$\lambda_2=1/3$
		$P:=((1,0),(3,1),(5,2))$	1		
2	1/3	$P:=((1,1),(3,2),(4,3),(5,4))$	2	$Y_2=(27,25)$	$\lambda_3=3/8$
		$P:=((1,0),(2,2),(5,3))$	1		
3	3/8	$P:=((1,0),(2,2),(5,3))$	2	$Y_3=(30,20)$	$\lambda_4=1/2$
		$P:=((1,1),(3,2),(4,3),(5,4))$	1		
4	1/2	$P:=((1,0),(2,2),(5,3))$	2	$Y_4=(31,19)$	$\lambda_5=1$

set L and, for the forward arcs (1,2) and (1,3) at time $\theta=0$, node-time pairs (2,2) and (3,1) are added to the set L, labelled as: $\pi(2,2):=2$, $\tau(2,2):=1$, $\pi(3,1):=2$, $\tau(3,1):=2$ and $p(2,2):=(1,0)$, $p(3,1):=(1,0)$ are set. Then the next node-time pair, (1,1) is removed from the set L and, for the forward arc (1,3) at time $\theta=1$, the node-time pair (3,2) is are added to the set L, labelled as: $\pi(3,2):=2$, $\tau(3,2):=2$ and $p(3,2):=(1,1)$. Since for the arc (1,2) at time $\theta=1$, the transit time $\theta_1+h(1,2;\theta_1)=1+3=4$ there will be no path which connects the node-time pair (2,4) to the sink node within the time horizon T = 4. The same thing will happen for the source node at all the other time values: $\theta=2,3,4$. Since the set of candidate node-time pairs, in increasing ordered of their corresponding distance labels, is now $L:=\{(2,2),(3,1),(3,2)\}$, the first one is removed and, for the forward arcs (2,4) and (2,5) at time $\theta=2$, node-time pairs (4,3) and (5,3) are added to the set L and labelled as: $\pi(4,3):=9$, $\tau(4,3):=-4$, $\pi(5,3):=11$ and $\tau(5,3):=-6$. The predecessor nodes are set to $p(4,3):=(2,2)$ and $p(5,3):=(2,2)$. Starting from node 3 at time $\theta=1$, *i.e.* from the node-time pair (3,1), the node-time pair (5,2) is labelled as $\pi(5,2):=9$, $\tau(5,3):=0$ and $p(5,2):=(3,1)$. As for the node-time pair (4,3), since $\pi(3,1)+\alpha_0(3,4;1)=6$ and $\pi(4,3)=9$, the new label is set to $\pi(4,3)=6$, $\tau(4,3):=4$ and the predecessor node is $p(4,3):=(3,1)$ instead of the previously stated value $p(4,3):=(2,2)$. Procedure **next_lambda**, invoked for relabeling node-time pair (4,3), computes the value $\lambda':=0+(6-9)/-4-4)=3/8$ which proves to be greater than $\lambda_0=0$ and smaller than $\lambda_1=1$ so that the next value of the parameter is set to $\lambda_{k+1}=\lambda_1:=3/8$.

Similarly, starting from node 3 at time $\theta=2$, *i.e.* from the node-time pair (3,2), since $\pi(3,2)+\alpha_0(3,4;2)=7$ and $\pi(4,3)=6$, the node-time pair (4,3) keeps its previously stated label but procedure **next_lambda**, still computes the value $\lambda':=0+(6-7)/(-2-4)=1/6$ which is greater than $\lambda_0=0$ and smaller than $\lambda_1=3/8$ so that the next value of the parameter is set to $\lambda:=1/6$. Finally, for the forward arc (4,5) at time $\theta=3$, the node-time pair (5,4) is labelled as $\pi(5,2):=7$, $\tau(5,3):=8$ and $p(5,4):=(4,3)$ is set.

Since the set L is empty, the minimum label of the sink node at all time values is computed as:

$\overline{\pi}(5):=\min\{\pi(5,0),\pi(5,1),\pi(5,2),\pi(5,3),\pi(5,4)\}$, *i.e.*

$\overline{\pi}(5):=\min\{\infty,\infty,9,11,7\}=7$ and $\overline{\tau}(5):=8$.

Procedure **next_lambda**, consecutively will compute the values $\lambda':=1/4$ and $\lambda':=3/14$, both being greater than the previously computed value of $\lambda_1:=1/6$.

Based on the predecessor vector, the shortest path $P:=((1,0),(3,1),(4,3),(5,4))$ is built and its residual capacity $r(P):=2$ is computed. The flow is augmented with $\delta:=\min\{v',r(P)\}=\min\{3,2\}=2$ units along this path, the residual network is correspondingly updated and the updated excess value υ' is set to $\upsilon':=3-2=1$.

Since $\upsilon'>0$, the algorithm reiterates in the updated residual network finding the shortest path

$$P:=((1,1),(3,2),(4,3),(3,1),(5,2))$$

with $r(P):=2$, $\delta:=\min\{1,2\}=1$ and $\upsilon':=0$, which makes the algorithm to stop. The first efficient solution f_0 in the decision space sends two units of flow along the path $P:=((1,0),(3,1),(4,3),(5,4))$ and one unit of flow along the path $P:=((1,1),(3,2),(4,3),(3,1),(5,2))$. The corresponding non-dominated extreme point in the objective space is $Y_0=(24,34)$.

***Iteration* 2**: With $k=1$, the algorithm computes the new parametric costs for $\lambda_1:=1/6$ and correspondingly finds the second efficient solution f_1, sending two units

of flow along the path $P := ((1,1),(3,2),(4,3),(5,4))$ and one unit of flow along the path $P := ((1,0),(3,1),(5,2))$, with the extreme point in the objective space $Y_1 = (25, 29)$, and $\lambda_2 := 1/3$.

The steps performed by the algorithm are described in **Table 5** and the objective space with the non-dominated extreme points is presented in **Figure 1(b)**.

6. References

[1] R. Ahuja, T. Magnanti and J. Orlin, "Network Flows. Theory, Algorithms and Applications," Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[2] J. A. Aronson, "A Survey of Dynamic Network Flows," *Annals of Operations Research*, Vol. 1, No.1, 1989, pp. 1-66. doi:10.1007/BF02216922

[3] W. Powell, P. Jaillet and A. Odoni, "Stochastic and Dynamic Networks and Routing," In: Ball, M. O., Magnanti, T. L., Monma, C. L. and Nemhauser G. L., Ed., Network Routing, Handbooks in Operations Research and Management Science, Chapter 3, North Holland, Amsterdam, The Netherlands, Vol. 8, 1995, pp. 141-295.

[4] N. Kamiyama, A. Takizawa, N. Katoh and Y. Kawabata, "Evaluation of Capacities of Refuges in Urban Areas by Using Dynamic Network Flows," Proceedings of the Eighth International Symposium on Operations Research and Its Applications, ORSC & APORC, Zhangjiajie, China, 2009, pp. 453-460.

[5] I. Chabini and M. Abou-Zeid, "The Minimum Cost Flow Problem in Capacitated Dynamic Networks," Annual Meeting of the Transportation Research Board, Washington, D.C., 2003, pp. 1-30.

[6] E. Nasrabadi and S. M. Hashemi, "Minimum Cost Time-Varying Network Flow Problems," *Optimization Methods and Software*, Vol. 25, No. 3, 2010, pp. 429-447. doi:10.1080/10556780903239121

[7] H. Lee and S. Pulat, "Bicriteria Network Flow Problems: Continuous Case," *European Journal of Operational Research*, Vol. 51, No. 1, 1991, pp. 119-126. doi:10.1016/0377-2217(91)90151-K

[8] S. Gass and T. Saaty, "The Computational Algorithm for the Parametric Objective Function," *Naval Research Logistics Quarterly*, Vol. 1, No. 1-2, 1955, pp. 39-45. doi:10.1002/nav.3800020106

[9] A. M. Geoffrion, "Solving Bicriterion Mathematical Programs," *Operations Research*, Vol. 15, No. 1, 1967, pp. 39-54. doi:10.1287/opre.15.1.39

[10] Y. P. Aneja and K. P. Nair, "Bicriteria Transportation Problem," *Management Science*, Vol. 25, No. 1, 1979, pp. 73-78. doi:10.1287/mnsc.25.1.73

[11] A. Skriver, "A Classification of Bicriterion Shortest Path (BSP) Algorithms," *Asia-Pacific Journal of Operational Research*, No. 17, 2000, pp. 199-212.

[12] X. Cai, D. Sha and C. Wong, "Time-Varying Network Optimization," Springer, New York, 2007.

