Algorithms for Time Dependent Bicriteria Shortest Path Problems *

Horst W. Hamacher †, Stevanus A. Tjandra
Fachbereich Mathematik, Universität Kaiserslautern, Germany

Abstract

In this paper we generalize the classical shortest path problem in two ways. We consider two - in general contradicting - objective functions and introduce a time dependency of the cost which is caused by a traversal time on each arc. The resulting problem, called time-dependent bicriteria shortest path problem (TdBiSP) has several interesting practical applications, but has not attained much attention in the literature. After reviewing relevant literature we develop two new algorithms in which one of them requires weaker assumptions as in previous research on the subject. Numerical tests show the superiority of the new algorithms.

Keywords: Multiple criteria analysis; Label setting algorithm; Label correcting algorithm

1 Introduction

Shortest path problems are one of the best studied network optimization problems (see e.g. Bertsekas [3] and Ahuja, et al [1]) with many applications, such as in transportations (see e.g. Ziliaskopoulos and Mahmassani [22], and Chabini [5]) and evacuation planning (see e.g. Kostreva and Wiecek [13], Yamada [21], and Getachew et al [9]). In this paper we consider **time dependent, bicriteria shortest path problems (TdBiSP)** which differ from the classical shortest path problems in two ways

- two, generally contradicting, objective functions instead of a single one
- time dependency of traversal of arcs, waiting at nodes, and cost function

TdBiSP is a valuable tool in several of the appliations mentioned above, since most of these processes are time dependent and more than one set of time dependent cost parameters are to be taken into account. Our main motivation is

*This research has been supported by Deutsche Forschungsgemeinschaft (DFG) grant spp-1126 "Algorithmik Grosser und Komplexer Netzwerke"

†Tel.: +49 631 205 2267; Fax: +49 631 ; E-mail: hamacher@mathematik.uni-kl.de

the application in evacuation modelling, where shortest paths represent evacuation routes. Obviously, these routes are in case of emergeny subject to changes over time (expressable as time varying costs) and different types of cost representing for instance speed or reliability may be of interest. A set of possible evacuation routes may also be put together to a dynamic flow which can be used to model a complete evacuation plan. Paths and cycles play in this context the role of building up and improving existing plans, respectively. In this situation cost may be negative as well as positive, a situation which will analyze later in this paper. An overview on the interrelation of dynamic network optimization and evacuation modeling can be found in Hamacher and Tjandra [10].

Without time dependency, there are several papers dealing with multicriteria shortest path problems. Hansen [11] introduced ten different types of bicriteria path problems. He also showed that one can construct a specific static network with attributes in which the number of Pareto optimal paths grows exponentially with the number of nodes in the network. Martins [14] developed a multiple labeling version of Djikstra's label setting algorithm to generate all Pareto shortest paths from the source node to every other node in the network (i.e. *one-to-all* shortest paths). Corley and Moon [7] used dynamic programming to solve the multicriteria shortest paths problem. Brumbaugh-Smith and Shier [4] proposed a label correcting with a multiple labeling for the bicriteria shortest path problem. They also proposed a linear time algorithm for merging two Pareto optimal sets.

On the other hand, there are various papers, dealing with time dependency, but with a single objective function. Here, the dynamic shortest path problem (also called as time-dependent shortest path) can be classified into fastest path and minimum cost path. In fastest path problem, the cost of an arc is the travel time of that arc. Its objective is then to seek for paths having minimum length with respect to time-dependent travel times. The minimum cost path problem seeks for paths having minimum length with respect to cost while considering the time needed to travel from one node to another. Hence, the fastest path problem is a particular case of the minimum cost path problem. The first paper dealing with the fastest path problem appears to be by Cooke and Halsey [6]. They extended Bellman's optimality principle (Bellman [2]) to obtain the fastest paths from every node in the network to one destination (i.e. sink) node, i.e. *all-to-one* fastest paths. The travel times on the arcs are defined to have positive integer values for every time period taken from a time-grid $\{t_0, t_0+1, \dots, t_0+M\}$. Here, the integer number M must be chosen in such a way that the problem is feasible. Dreyfus [8] proposed a modification of Djikstra's static shortest path algorithm to calculate the fastest path between two nodes for a given departure time. His algorithm works well only for the FIFO (First-In First-Out) network, that is a network having a property that for every its arc (i, j), a unit leaving node i earlier will not arrive at node j along (i, j) later than the one that leaving later. This FIFO property is also known as *non-overtaking* property in traffic disciplines. The complexity of Dreyfus's algorithm is the same as that of Djikstra. Ziliaskopoulos and Mahmassani [22] proposed label-correcting algorithms to calculate both *all-to-one* fastest paths and minimum cost paths for every possible departure times. Their algorithms do not require a FIFO networks. Similar algorithms were proposed by Wardell and Ziliaskopoulos [20]. Chabini [5] proposed backward label setting algorithms running from the time horizon T down to zero in a decreasing manner. His algorithms solves both fastest and minimum cost paths for all possible departure times without FIFO requirement. The labeling is done according to time steps t such that labels stay unchanged in time steps $t' > t$. In a continuous-time environment, several results on time-dependent shortest paths (i.e. fastest path or minimum cost path) can be mentioned here, including Orda and Rom [15, 16], Philpott and Mees [19], and Philpott [18]).

In the literature, multicriteria dynamic shortest path problems, as combination of the above two problem classes, has attained relatively few attention. Kostreva and Wiecek [13] generalized the result of Cooke and Halsey [6] to the multicriteria case. The cost function of arcs are assumed to be positive vector valued functions of time and may be discontinuous. A backward dynamic programming is developed to generate all Pareto minimum cost paths leading from every node in the network to the sink node. A forward dynamic programming is also discussed by these authors to generate all Pareto minimum cost paths from the source to every other nodes in the network under FIFO and nondecresing arc costs assumptions. Getachew, et al [9] generalize the results of Kostreva and Wiecek by replacing their nondecreasing arc cost assumption by lower and upper bounds of the cost (for the forward dynamic programming) and by relaxing time grid assumption. To the best of our knowledge, these two papers are the only ones which have been published in the area of multicriteria time-dependent shortest path problems.

In this paper we will develop two algorithms solving TdBiSP. While the first algorithm deals only with the non-negative data, the second one allows the arcs of the network to have negative travel times and costs. The possibility to wait (or park) at a node before departing on outgoing arc, in order to keep the total cost low, is also taken into account. In the next section, we formally introduce TdBiSP. In Section 3 we classify the problem as NP-hard, show that Belman's optimality principle holds for the all-to-one problem, but not the one-to-all problem and introduce the concept of negative dynamic cycles. Section 4 deals with the necessary data structures for our labels, which take into consideration the time and bicriteria environment. In Section 5 we assume non-negative data, work out details of the algorithm of Kostreva and Wiecek [13] and present a new, label setting algorithm. The non-negativity assumption is dropped in the subsequent Section 6, where a label correcting algorithm is developed which can be used to detect negative dynamic cycles, or solve TdBiSP if such cycles do not exist. Computational results on several examples on randomly generated networks in Section 7 conclude this paper.

2 The Time Dependent Bicriteria Shortest Path Problem

A discrete-time dependent network $G = (N, A, T)$ is a directed graph, where N is the set of nodes, A is the set of directed arcs, and T is a finite time horizon of interest discretized into the set $\{0, \dots, T\}$. We denote by n and m the cardinality of N and A, respectively. Moreover, it is assumed that G has a single sink d. Each arc (i, j) carries a travel time function $\lambda_{ij}(t)$ with

$$\lambda : A \times \{0, \dots, T\} \to \mathbb{Z}^m$$

and a cost (vector) function $\begin{pmatrix} c^1 \\ c^2 \end{pmatrix}$ of two criteria components with

$$c^k : A \times \{0, \dots, T\} \to \mathbb{R}^m, \quad k = 1, 2$$

where the superscript k corresponds to the cost function of criteria k. The travel time $\lambda_{ij}(t)$ defines the time period needed to traverse the arc (i, j), departing from node i at time t. This travel time is defined upon entering an arc, and it is assumed to be constant for the duration of travel along that arc. This model of travel time is known as a *frozen arc model* [15]. Waiting in any node $i \in N - \{d\}$ at any time $t \in \{0, \dots, T-1\}$ is described by $w_i(t)$, the maximum allowable waiting time, and $h_i(t)$, the corresponding waiting cost. The value of parameter $w_i(t)$ is decided by the current time t and not by the arrival time at the node. Therefore, it has a value of either zero or one.

We define a dynamic path and dynamic cycle, respectively, as follows.

Definition 2.1 (Dynamic path) *A dynamic $j_1 - j_l$ path $P_{j_1 j_l}(t_1)$ is a sequence of node-time pairs (NTPs) from the node j_1 to the node j_l that is ready at node j_1 at time $t_1 \in \{0, \dots, T\}$, as given by*

$$P_{j_1 j_l}(t_1) = \{j_1(t_1, t_1^{'}), \dots, j_q(t_q, t_q^{'}), \dots, j_l(t_l, t_l^{'})\}, \quad t_q, t_q^{'} \in \{0, \dots, T\}, \quad q = 1, \dots, l \tag{1}$$

where $t_{q+1} = t_q^{'} + \lambda_{j_q j_{q+1}}(t_q^{'})$, $q = 1, \dots, l-1$ and define $t_l = t_l^{'}$.
For any NTP $j_q(t_q, t_q^{'})$, the first time parameter t_q denotes the ready time at node j_q, and the second one $t_q^{'}$ denotes the departure time from node j_q. A ready time t_q also defines the arrival time at node j_q from the previous node j_{q-1}, for $q = 2, \dots, l$.

Figure 2 illustrates Definition 2.1.

If $t \in \{0, \dots, T\}$ is a ready time of node i and $t^{'}$ is a departing time from node i with $t^{'} \geq t$, then the corresponding total waiting cost at node i, denoted by $H_i(t, t^{'})$, is given by

$$H_i(t, t^{'}) = \sum_{t^{''}=t+1}^{t^{'}} h_i(t^{''})$$

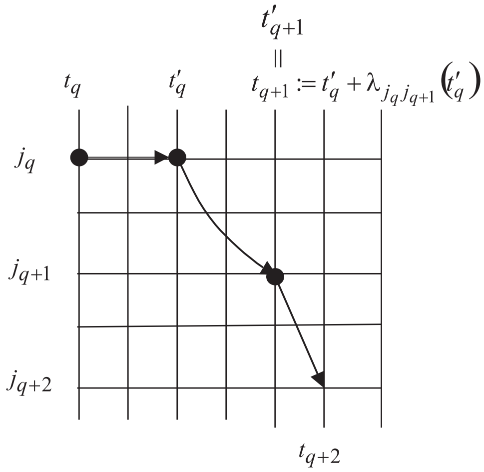


Figure 1: Dynamic Path

We denote by $\mathbb{P}_{j_1 j_l}(t_1)$ the set of all paths from node j_1 with ready time $t_1 \in \{0, \dots, T\}$ and reaching the node j_l within the time horizon T. We also denote by $\mathbb{P}_{j_1 j_l}$ the union of $\mathbb{P}_{j_1 j_l}(t_1)$ for all $t_1 \in \{0, \dots, T\}$.

Definition 2.2 *The length of dynamic path $P_{j_1 j_l}(t_1)$ with respect to the travel times and waiting times is given by*

$$\lambda(P_{j_1 j_l}(t_1)) := t_l^{'} - t_1 \tag{2}$$

Definition 2.3 (Dynamic cycle) *A dynamic $j_1 - j_l$ path $P_{j_1 j_l}(t_1)$ defined by* (1) *is a dynamic cycle if $j_1 = j_l$, $t_1 = t_l$ and $t_1^{'} = t_l^{'}$.*

It should be noted that dynamic cycle may exist, since negative travel times are allowed. Moreover, a dynamic $j_1 - j_l$ path $P_{j_1 j_l}(t_1)$ is called *simple* if it does not contain any dynamic cycle.

To define the cost of a path, it is useful to define the following operation. For $\begin{pmatrix} g^1 \\ g^2 \end{pmatrix} \in \mathbb{R}^2$ and $e \in \mathbb{R}$ define

$$\begin{pmatrix} g^1 \\ g^2 \end{pmatrix} \oplus e = \begin{pmatrix} g^1 + e \\ g^2 + e \end{pmatrix} \tag{3}$$

Let $c : \mathbb{P}_{j_1 j_l} \to \mathbb{R}^2$. The total cost of a $j_1 - j_l$ path $P_{j_1 j_l}(t_1) \in \mathbb{P}_{j_1 j_l}(t_1)$ is defined by

$$c(P_{j_1 j_l}(t_1)) := \Bigg((\begin{array}{c} c^1(P_{j_1 j_l}(t_1)) \\ c^2(P_{j_1 j_l}(t_1)) \end{array} \Bigg) = \sum_{q=1}^{l-1} \begin{pmatrix} c^1_{j_q j_{q+1}}(t_q^{'}) \\ c^2_{j_q j_{q+1}}(t_q^{'}) \end{pmatrix} \oplus H_{j_q}(t_q, t_q^{'}) \tag{4}$$

Using this cost definition, we give a definition of a negative dynamic cycle as follows.

Definition 2.4 (Negative dynamic cycle) *A dynamic cycle $P_{j_1 j_l}(t_1)$ is a negative dynamic cycle if at least one component of $c(P_{j_1 j_l}(t_1))$ has a negative value.*

For every node $i \in N$ and every ready time $t \in \{0, \dots, T\}$, we want to find a shortest dynamic path from i to a single sink d with respect to cost vector c. But, in general, there exists no dynamic path that simultaneously minimizes $c^k(P_{id}(t))$ for both $k = 1, 2$. Nevertheless a privileged set of paths called the set of *Pareto optimal* (*efficient* or *nondominated*) paths can be determined. To describe a Pareto optimal path, we need the notion of a vector ordering.

Definition 2.5 (Vector ordering) *Let $z_1 = \begin{pmatrix} z_1^1 \\ z_1^2 \end{pmatrix}$ and $z_2 = \begin{pmatrix} z_2^1 \\ z_2^2 \end{pmatrix}$ be two vectors in $\mathbb{R}^2$.*

- *$z_1 = z_2$ if and only if $z_1^k = z_2^k$ for all $k = 1, 2$.*
- *$z_1 \leq z_2$ if and only if $z_1^k \leq z_2^k$ for all $k = 1, 2$.*
- *$z_1 < z_2$ if and only if $z_1^k \leq z_2^k$ for all $k = 1, 2$ and there exists $k \in \{1, 2\}$ such that $z_1^k < z_2^k$.*

Definition 2.6 (Pareto optimal path and minimal complete set)

(a) *Let $P_{id}(t)$ and $P'_{id}(t)$ be two paths in $\mathbb{P}_{id}(t)$. We say $P_{id}(t)$ dominates $P'_{id}(t)$ if and only if*

$$c(P_{id}(t)) < c(P'_{id}(t))$$

(b) *A path $P_{id}(t) \in \mathbb{P}_{id}(t)$ is called Pareto optimal (PO) or nondominated if there is no other path $P'_{id}(t) \in \mathbb{P}_{id}(t)$ dominating $P_{id}(t)$.*

(c) *Two PO-paths $P_{id}(t)$ and $P'_{id}(t)$ in $\mathbb{P}_{id}(t)$ are said equivalent if and only if $c(P'_{id}(t)) = c(P_{id}(t))$. Otherwise, they are said nonequivalent.*

(d) *A complete set $\overline{PO}(\mathbb{P}_{id}(t)) \subset \mathbb{P}_{id}(t)$ of PO-paths is a set such that any path $P_{id}(t) \notin \overline{PO}(\mathbb{P}_{id}(t))$ is either dominated or equivalent to at least one PO-path $P'_{id}(t) \in \overline{PO}(\mathbb{P}_{id}(t))$.*

(e) *A complete set is minimal if and only if it does not have two equivalent PO-paths. We then denote by $PO(\mathbb{P}_{id}(t))$ the minimal complete set of all Pareto optimal paths from node i to the sink node d with ready time $t \in \{0, \dots, T\}$ at node i.*

3 Problem Characteristics

3.1 Complexity

Looking at the problem complexity, as in the case of bicriterion static path problem (see [11]), Example 3.1 shows a specific network with time dependent attributes in which the number of PO-paths grow exponentially with the number of nodes in the network. Therefore, in the worst case, this problem is intractable and any algorithm for generating all Pareto dynamic paths is thus doomed to an exponential computation time.

Example 3.1
Consider the network in Figure 2.

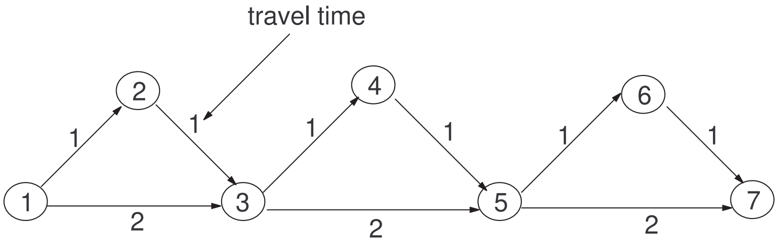


Figure 2: Example network with exponential number of Pareto optimal paths

We assign the following two cost criteria to each arc of the network.

$$c_{ij}(t) = \begin{cases} \begin{pmatrix} 2^i(t+1) \\ 2^{i-1}(t+1) \end{pmatrix} & \text{, if } j = i+1 \text{ and } i \text{ is odd} \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \text{, if } j = i+1 \text{ and } i \text{ is even} \\ \begin{pmatrix} 2^{i-1}(t+1) \\ 2^i(t+1) \end{pmatrix} & \text{, if } j = i+2 \text{ and } i \text{ is odd} \end{cases} \quad ;\ t \in \{0, \dots, T\}$$

Let us assume that $w_i(t) = 0$ for any node $i \in N$ and any time $t \in \{0, \dots, T\}$, i.e. no waiting is allowed for any node. By this cost definition, there are $2^{\frac{n-1}{2}} = 8$ PO-paths from node 1 to node 7 for every ready time $0 \leq t_0 \leq T-6$, as shown in Table 3.1. □

3.2 Principle of optimality

In the case of static shortest path problems, the principle of optimality says that the subsets of the optimal solution are also optimal(see e.g. [2]). Concerning the multicriteria dynamic shortest path problem, we formalize the principle optimality as follows.

Definition 3.1 (Principle of optimality in a forward direction)
Let $P_{sj}(t) \in PO(\mathbb{P}_{sj}(t))$. The forward principle of optimality holds for $P_{sj}(t)$ if and only if for each intermediate node i of $P_{sj}(t)$, the sub-path $P_{si}(t)$ is also in $PO(\mathbb{P}_{si}(t))$.

$P_{17}(t_0),\ 0 \leq t_0 \leq T-6$	$c(P_{17}(t_0))$
$\{1(t_0,t_0), 2(t_0+1,t_0+1), 3(t_0+2,t_0+2), 4(t_0+3,t_0+3), 5(t_0+4,t_0+4), 6(t_0+5,t_0+5)\}, 7(t_0+6,t_0+6)\}$	$\begin{pmatrix} 186+42t_0 \\ 93+21t_0 \end{pmatrix}$
$\{1(t_0,t_0), 2(t_0+1,t_0+1), 3(t_0+2,t_0+2), 4(t_0+3,t_0+3), 5(t_0+4,t_0+4), 7(t_0+6,t_0+6)\}$	$\begin{pmatrix} 106+26t_0 \\ 173+37t_0 \end{pmatrix}$
$\{1(t_0,t_0), 2(t_0+1,t_0+1), 3(t_0+2,t_0+2), 5(t_0+4,t_0+4), 7(t_0+6,t_0+6)\}$	$\begin{pmatrix} 94+22t_0 \\ 185+41t_0 \end{pmatrix}$
$\{1(t_0,t_0), 2(t_0+1,t_0+1), 3(t_0+2,t_0+2), 5(t_0+4,t_0+4), 6(t_0+5,t_0+5), 7(t_0+6,t_0+6)\}$	$\begin{pmatrix} 174+38t_0 \\ 105+25t_0 \end{pmatrix}$
$\{1(t_0,t_0), 3(t_0+2,t_0+2), 5(t_0+4,t_0+4), 6(t_0+5,t_0+5), 7(t_0+6,t_0+6)\}$	$\begin{pmatrix} 173+37t_0 \\ 106+26t_0 \end{pmatrix}$
$\{1(t_0,t_0), 3(t_0+2,t_0+2), 5(t_0+4,t_0+4), 7(t_0+6,t_0+6)\}$	$\begin{pmatrix} 93+21t_0 \\ 186+42t_0 \end{pmatrix}$
$\{1(t_0,t_0), 3(t_0+2,t_0+2), 4(t_0+3,t_0+3), 5(t_0+4,t_0+4), 7(t_0+6,t_0+6)\}$	$\begin{pmatrix} 105+25t_0 \\ 174+38t_0 \end{pmatrix}$
$\{1(t_0,t_0), 3(t_0+2,t_0+2), 4(t_0+3,t_0+3), 5(t_0+4,t_0+4), 6(t_0+5,t_0+5), 7(t_0+6,t_0+6)\}$	$\begin{pmatrix} 185+41t_0 \\ 94+22t_0 \end{pmatrix}$

Table 1: Pareto optimal paths and their associated values of Example 3.1

Definition 3.2 (Principle of optimality in a backward direction)
Let $P_{id}(t) \in PO(\mathbb{P}_{id}(t))$. The backward principle of optimality holds for $P_{id}(t)$ if and only if for each intermediate NTP $j(t_j, t_j^{'})$ of $P_{id}(t)$, the sub-path $P_{jd}(t_j)$ is also in $PO(\mathbb{P}_{jd}(t_j))$.

Definition 3.3 (e.g. [17]) *An arc (i,j) is said to have the Cost Consistency (CC) property if leaving node i earlier along (i,j) does not cost more than leaving later, i.e.*

$$c_{ij}(t) \leq c_{ij}(t^{'}) + \sum_{t^{''}=t+1}^{t^{'}} h_i(t^{''}-1),\ \forall t < t^{'}$$

A dynamic network $G = (N, A, T)$ is said to be a CC network when all its arcs have the CC property.

The following theorem shows that the backward principle of optimality holds, even if FIFO or cost consistency assumptions do not hold.

Theorem 3.1 *Let $G = (N, A, T)$ be a dynamic network with positive parameters. The backward principle of optimality holds for any Pareto optimal path in G.*

Proof:
Let $P_{id}(t) \in PO(\mathbb{P}_{id}(t))$ with $j(t_j, t_j^{'}) \in P_{id}(t)$. Assume that $P_{jd}(t_j) \subset P_{id}(t)$ is not a Pareto optimal, i.e. there exists some path $P_{jd}^{'}(t_j)$ such that

$$c(P_{jd}^{'}(t_j)) < c(P_{jd}(t_j)) \tag{5}$$

Let $P_{ij}(t)$ be the subpath of $P_{id}(t)$. By concatenating $P_{ij}(t)$ with $P_{jd}(t_j)$ and $P_{jd}^{'}(t_j)$, we obtain two paths $P_{id}(t)$ and $P_{id}^{'}(t)$ such that their total cost is, respectively

$$c(P_{id}(t)) = c(P_{ij}(t)) + c(P_{jd}(t_j))$$

and

$$c(P_{id}^{'}(t)) = c(P_{ij}(t)) + c(P_{jd}^{'}(t_j))$$

Condition (5) implies

$$c(P_{id}^{'}(t)) < c(P_{id}(t))$$

contradicting the fact that $P_{id}(t)$ is a Pareto optimal path. Hence, $P_{jd}(t_j)$ must be a Pareto optimal path. ■

In contrast, one can build a dynamic network, even if FIFO and CC properties are satisfied, such that the forward principle of optimality does not hold.

Example 3.2
Consider the dynamic network $G = (N, A, T = 8)$ shown in Figure 3. Node 0 is the source node and node 3 is the sink node. Clearly each arc in G has a FIFO property and every cost component has a CC property. The path $P_{03}(0) = \{0(0,0), 2(1,1), 3(2,2)\}$ is a Pareto optimal path from node 0 to node 3 with ready time $t = 0$. Its total cost is

$$c(P_{03}(0)) = \begin{pmatrix} 9 \\ 11 \end{pmatrix}$$

This path dominates $P_{03}^{'}(0) = \{0(0,0), 1(3,3), 2(4,4), 3(5,5)\}$ which has total cost

$$c(P_{03}^{'}(0)) = \begin{pmatrix} 12 \\ 15 \end{pmatrix}$$

The path $P_{02}^{'}(0) = \{0(0,0), 1(3,3), 2(4,4)\}$ is a Pareto optimal path from node 0 to node 2 with ready time $t = 0$. Its total cost is

$$c(P_{02}^{'}(0)) = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$$

$$\lambda_{01}(t) = 3,\ t \geq 0\ ;\ \lambda_{02}(t) = \lambda_{12}(t) = \lambda_{23}(t) = 1,\ t \geq 0$$
$$w_i(t) = 0,\ i \in N,\ t \geq 0$$
$$c_{01}(t) = c_{12}(t) = \begin{pmatrix} 1 \\ 2 \end{pmatrix},\ t \geq 0$$
$$c_{02}(t) = \begin{pmatrix} 5 \\ 6 \end{pmatrix},\ t \geq 0$$
$$c_{23}(t) = \begin{pmatrix} 2t+2 \\ 2t+3 \end{pmatrix},\ t \geq 0$$

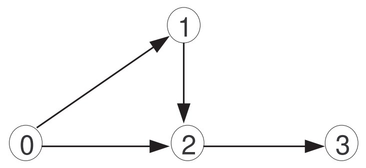


Figure 3: A network for Example 3.2

This path dominates $P_{02}(0) = \{0(0,0), 2(1,1)\}$ which has total cost

$$c(P_{02}(0)) = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

We see that $P_{02}(0)$ is a subset of a Pareto optimal path $P_{03}(0)$, but it is dominated. Hence, the forward principle of optimality does not hold. □

If the principle of optimality is not valid, optimal subpaths as partial results can not be used to obtain the optimal paths. Therefore, one cannot use the forward label setting-type algorithm to find Pareto optimal paths. However, Theorem 3.1 guarantees that we can use the label setting principles in a backward direction to find the *all-to-one* Pareto optimal paths. This will be worked out in more detail in Section 5.

3.3 Negative dynamic cycle

We allow in this subsection positive as well as negative travel times. In case of positive travel times, a dynamic path may pass through the same node several times but with increasing arrival times (i.e. the path must pass through different node-time pairs). Therefore, there will be no negative dynamic cycle. Furthermore, since T is finite, the cost of this path is also finite, disregarding the fact that the cost may also be negative. But, if the network has negative travel times, then there may exist some negative dynamic cycles within $\{0, \ldots, T\}$. The following example illustrates these phenomena.

Example 3.3
Look at the network shown in Figure 3.3.
Let $T = 17$ and assume that no waiting is allowed. Consider a dynamic path $P_{17}(0)$ from node 1 to node 7 with ready time $t_0 = 0$.

$$\begin{aligned} P_{17}(0) \;=\; & \{1(0,0), 2(1,1), 4(4,4), 6(5,5), 5(6,6), 2(7,7), 4(10,10), \\ & 6(11,11), 5(12,12), 2(13,13), 4(16,16), 7(17,17)\} \end{aligned}$$

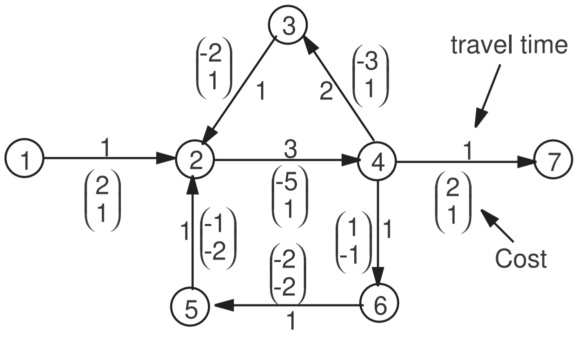


Figure 4: Network with finite Pareto dynamic cycle

The total cost of path $P_{17}(0)$ is

$$c(P_{17}(0)) = \begin{pmatrix} -15 \\ -5 \end{pmatrix}$$

This path is a PO-path and passes through the same nodes 2 and 4 three times, nodes 5 and 6 two times, with different departure and arrival times. Hence, $P_{17}(0)$ does not contain any dynamic cycle. Other Pareto optimal paths in $\mathbb{P}_{17}(0)$ are

$$\begin{aligned} P'_{17}(0) &= \{1(0,0), 2(1,1), 4(4,4), 3(6,6), 2(7,7), 4(10,10), 3(12,12), \\ &\quad 2(13,13), 4(16,16), 7(17,17)\} \end{aligned}$$

with value $c(P'_{17}(0)) = \begin{pmatrix} -21 \\ 9 \end{pmatrix}$ and

$$\begin{aligned} P''_{17}(0) &= \{1(0,0), 2(1,1), 4(4,4), 3(6,6), 2(7,7), 4(10,10), 6(11,11), \\ &\quad 5(12,12), 2(13,13), 4(16,16), 7(17,17)\} \end{aligned}$$

with value $c(P''_{17}(0)) = \begin{pmatrix} -18 \\ 2 \end{pmatrix}$.

Now, if we change the travel time of arcs $(4,6), (6,5), (5,2)$ to -1, then for any t with $t, t+3 \in \{0, \ldots, T\}$ we will obtain the following negative dynamic cycle

$$\{2(t,t), 4(t+3,t+3), 6(t+2,t+2), 5(t+1,t+1), 2(t,t)\}$$

with cost $\begin{pmatrix} -7 \\ -4 \end{pmatrix} < \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $\square$

4 Label Structure

We denote by $\pi_i(t)$ a label associated with node i at time t with the following components (see Figure 5) :

- a tuple of reals π^1, π^2, corresponding to the total cost with respect to the two cost criteria,
- a tuple of integers t_{ready}, t_{depart}, corresponding to the ready time and departure time, respectively,
- a pointer *succ_ptr* points to the Pareto optimal label of the successor node.

The tuple of reals, tuple of integers, and pointer, are then called the *cost component*, *time component*, and *successor pointer*, respectively, of the label $\pi_i(t)$. Hence, $\pi_i(t)$ is represented by

$$\pi_i(t) := \left(\begin{pmatrix} \pi^1 \\ \pi^2 \end{pmatrix}; (t_{ready}, t_{depart}); succ\_ptr \right) \tag{6}$$

We denote by $cost(\pi_i(t))$ the cost component of label $\pi_i(t)$. The need to store the time components makes the structure of a label $\pi_i(t)$ more complex than that of the static path problem.

We denote by $\Pi_i(t)$ the set of labels of node i at time t. This set is represented as a singly linked list having $\mid \Pi_i(t) \mid$ cells and each cell corresponds to the label $\pi_i(t) \in \Pi_i(t)$. Each cell will have six fields as the amount of information we wish to store. Five data fields will store the five components of $\pi_i(t)$ and the sixth field will store a pointer to the next cell in the list. We denote by *next_ptr* the pointer to the next cell in the list. If a cell happens to be the last cell in the list, by convention we set its link to $\emptyset$.

Example 4.1
Consider Example 3.3. Figure 5 shows a list representation of $\Pi_6(11)$. □

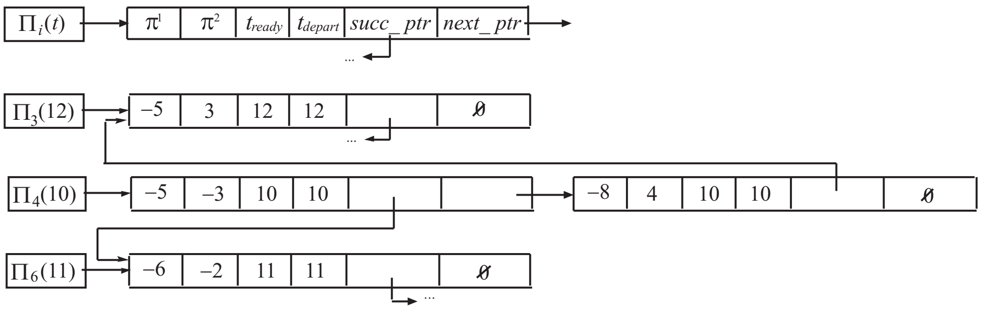


Figure 5: Label structure

5 Algorithm for All Ready Times with Non-Negative Data

In this section, we first discuss the algorithm fof Kostreva and Wiecek [13]. Next, we will discuss our proposed backward label setting algorithm. With non-negative data, we mean the cost components may have zero or positive value, but the travel time must be positive.

5.1 Backward Dynamic Programming

Kostreva and Wiecek [13] developed a backward dynamic programming approach to generate all Pareto optimal paths leading from every node in the network to the sink node. In order to update the set of Pareto labels of node i at time $t_{ready} \in \{0, \ldots, T\}$, the algorithm considers the labels of all node j with $(i, j) \in A$ at every $t_{arrival} \in \{0, \ldots, T\}$ in which $t_{arrival} = t_{depart} + \lambda_{ij}(t_{depart})$. The value of t_{depart} is determined by the value of waiting time at node i starting at t_{ready}. The backward dynamic programming in [13] does not consider the waiting time, but it can easily be extended to that case. Here, instead of modeling the waiting by a self-loop as suggested by the authors, we consider it directly in the labeling process as follows. Since it is assumed that the maximum waiting time has a memory-less property, the value of t_{depart} runs over all possible departure times in the set $Depart_i(t_{ready})$ defined by

$$\begin{aligned} Depart_i(t_{ready}) \quad := \quad & \{t_{ready}\} \cup \\ & \{t^{'} \ : \ t_{ready} + 1 \leq t^{'} \leq t \leq T \text{ with} \\ & \prod_{t^{''} = t_{ready}}^{t-1} w_i(t^{''}) > 0 \text{ and } w_i(t) = 0\} \end{aligned} \tag{7}$$

The backward dynamic programming approach is summarized as follows. For any set $\Pi_j(t)$ of labels, we denote by $VMIN(\Pi_j(t))$ the minimization over the associated cost components of all members of the label set $\Pi_j(t)$. This process yields the set of labels with associated Pareto optimal costs. Let $\Pi_i(t_{ready})^{(l)}$ be the set of labels of all Pareto optimal $i - d$ paths with ready time t_{ready} of at most l arcs. The set $\Pi_i(t_{ready})^{(l+1)}$ for any $t_{ready} \in \{0, \ldots, T\}$ is obtained by

$$\Pi_i(t_{ready})^{(l+1)} = \begin{cases} VMIN\big(\Pi_j(t_{arrival})^{(l)} + c_{ij}(t_{depart}) \oplus \\ \qquad\qquad H_i(t_{ready}, t_{depart})\big), \ i \in N - \{d\} \\ \left\{\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} ; (t_{ready}, t_{ready}); \emptyset\right)\right\}, \ i = d \end{cases} \tag{8}$$

where $t_{arrival} = t_{depart} + \lambda_{ij}(t_{depart})$, for all $t_{depart} \in Depart_i(t_{ready})$.
The operation

$$\Pi_j(t_{arrival})^{(l)} + c_{ij}(t_{depart}) \oplus H_i(t_{ready}, t_{depart})$$

sums the associated cost component of every label in $\Pi_j(t_{arrival})^{(l)}$ with the vector $c_{ij}(t_{depart}) \oplus H_i(t_{ready}, t_{depart})$. This operation also sets the associated time component of every label in the set $\Pi_i(t_{ready})^{(l+1)}$ to (t_{ready}, t_{depart}) and the associated successor pointer to node j. The iteration continues until $\Pi_i(t_{ready})^{(l)}$ converges to $\Pi_i(t_{ready})$ for all $i \in N$ and $t_{ready} \in \{0, \ldots, T\}$. The complete algorithm is summarized in Algorithm 5.1.

To solve the vector minimization problem in Step 2 of Algorithm 5.1, we sort the

Algorithm 5.1 [13]

INPUT	Network $G = (N, A, T)$, $w(t), c(t)$, and $h(t)$.
OUTPUT	$PO(\mathbb{P}_{id}(t))$, $\forall i \in N$, $\forall t \in \{0, \ldots, T\}$.
0	Modify the cost vector $c_{ij}(t)$, $\forall t \in \{0, \ldots, T\}$ as follows. $$c_{ij}(t) = \begin{cases} \begin{pmatrix} c^1_{ij}(t) \\ c^2_{ij}(t) \end{pmatrix} & , t + \lambda_{ij}(t) \le T \\ \begin{pmatrix} \infty \\ \infty \end{pmatrix} & , t + \lambda_{ij}(t) > T \end{cases} \quad ; \ \forall (i,j) \in A$$
1	Assign the initial label to each node for all $t \in \{0, \ldots, T\}$ $$\Pi_i(t)^{(0)} = \begin{cases} \left\{ \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} ; \ (t,t); \emptyset \right) \right\} & , i = d \\ \emptyset & , \text{otherwise} \end{cases}$$ Set $l := 0$.
2	Calculate the new set $\Pi_i(t_{ready})^{(l+1)}$, $\forall i \in N$, $\forall t_{ready} \in \{0, \ldots, T\}$, as determined by (8).
3	If $\Pi_i(t_{ready})^{(l+1)} = \Pi_i(t_{ready})^{(l)}$, $\forall i \in N$, $\forall t_{ready} \in \{0, \ldots, T\}$, then go to Step 4. Otherwise, set $l := l + 1$ and go to Step 2.
4	For any $i \in N$ and any $t \in \{0, \ldots, T\}$, output $PO(\mathbb{P}_{id}(t))$ as obtained by forwardtracking the successor pointers of $\Pi_i(t)$. Terminate the algorithm.

set of labels $\Pi_j(t_{arrival})^{(l)} + c_{ij}(t_{depart}) \oplus H_i(t_{ready}, t_{depart})$ according to the first criteria. If we denote by b the cardinality of this set of labels, then the sorting process can be done in $\mathcal{O}(b \ log \ b)$ by applying *mergesort technique.* Maintaining the set of labels as a linked list will give an advantage to the implementation of mergesort technique, since there will be no demand for an extra space $\mathcal{O}(b)$ as if this sorting technique is applied on arrays. On the set of sorted labels (with respect to the first criteria), the minimum vectors can be obtained in $\mathcal{O}(b)$. Therefore, the vector minimization problem in Step 2 of Algorithm 5.1 can be solved in $\mathcal{O}(b \ log \ b)$.

5.2 Backward Label Setting Algorithm

Since all data are non-negative, no negative dynamic cycle will exist. We therefore in this subsection, extend the idea of Djikstra's label-setting static shortest path. Theorem 3.1 guarantees that the labeling process can be done in a *backward direction from the sink node d to all other nodes in the network.*

We consider, over the set of all labels associated with each node $i \in N$ at each ready time $t_{ready} \in \{0, \ldots, T\}$, the set of *temporary* and *permanent labels* denoted by $\Pi_i(t_{ready})_{prm}$ and $\Pi_i(t_{ready})_{tmp}$, respectively. The set $\Pi_i(t_{ready})_{prm}$ contains only labels having nonequivalent Pareto optimal cost components. The corresponding dynamic paths from this node i with ready time t_{ready} can be found by tracing its successor pointers (i.e. *succ_ptr*). At any intermediate step of the algorithm, the permanent labels remain unchanged, but the temporary labels can be changed or deleted. Initially, for any node i and any time t_{ready}, we define $\Pi_i(t_{ready})_{prm}$ by an empty set. Since we assume that no arc leaves the sink node d, we assign the sink node d at any time $t_{ready} \in \{0, \ldots, T\}$, a temporary label having cost component zero, and every other node i an empty set, i.e.

$$\Pi_i(t_{ready})_{tmp} \quad := \quad \begin{cases} \left\{ \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}; (t_{ready}, t_{ready}); \emptyset \right) \right\} & , i = d \\ \emptyset & , \text{otherwise} \end{cases} , \qquad t_{ready} \in \{0, \ldots, T\} \tag{9}$$

At each iteration, a label is removed from the set of temporary labels and put into the set of permanent labels. This means that a label is set permanently in each iteration. Any label $\pi_i(t_{ready})$ in $\Pi_i(t_{ready})_{prm}$ determines a dynamic path from i with ready time t_{ready} that contains only node-time pairs $j^{'}(t_{ready_{j^{'}}}, t_{depart_{j^{'}}})$ having labels in $\Pi_{j^{'}}(t_{ready_{j^{'}}})_{prm}$ as internal nodes.

Suppose that a label $\pi_j^*(t_{arrival}) \in \Pi_j(t_{arrival})_{tmp}$ is chosen as the *pivot label* to be put in $\Pi_j(t_{arrival})_{prm}$, i.e.

$$\Pi_j(t_{arrival})_{prm} := \Pi_j(t_{arrival})_{prm} \cup \{\pi_j^*(t_{arrival})\}$$

This pivot label is then used to generate the new labels (as candidate of temporary labels) for all nodes i at time t_{ready} that can reach node j at time $t_{arrival}$. Let $\hat{\pi}_i(t_{ready})$ be the candidate for the temporary label of node i at ready time t_{ready}. We define

$$\begin{aligned} \hat{\pi}_i(t_{ready}) \quad := \quad & \pi_j^*(t_{depart} + \lambda_{ij}(t_{depart})) + (c_{ij}(t_{depart}) \oplus H_i(t_{ready}, t_{depart})), \\ & (i,j) \in A, \;\; t_{depart} + \lambda_{ij}(t_{depart}) = t_{arrival}, \\ & t_{depart} \in Depart_i(t_{ready}) \end{aligned} \tag{10}$$

This label $\hat{\pi}_i(t_{ready})$ is then merged with $\Pi_i(t_{ready})_{tmp}$. Executing this step means deleting all labels representing dominated dynamic paths from i with ready time t_{ready} to the sink. The algorithm stops when $\Pi_i(t_{ready})_{tmp} = \emptyset$ for all $i \in N$ and all $t_{ready} \in \{0, \ldots, T\}$.

Let us define the following set operation *Merge* to obtain the minimal complete set of $B_1 \cup B_2$ with B_1 and B_2 two set of labels. We denote by $cost(v)$ the cost component of $v \in B_1 \cup B_2$. For any B_1 and B_2 we define

$$\begin{aligned} Merge(B_1, B_2) \quad = \quad & B_1 \cup B_2 - \{v \in B_1 \cup B_2 \; : \; cost(u) \leq cost(v) \text{ for some} \\ & \qquad u \neq v \text{ with } u, v \in B_1 \cup B_2\} \end{aligned} \tag{11}$$

where relation $\leq$ is given by Definition 2.5. This merge operation includes the process of deleting the corresponding time and pointer components of the dominated cost components. By (11), $Merge(B_1, B_2)$ contains only the nonequivalent Pareto optimal labels in $B_1 \cup B_2$. Since the set $\{\hat{\pi}_i(t_{ready})\}$ is a singleton, it is obvious that $Merge(\{\hat{\pi}_i(t_{ready})\}, \Pi_i(t_{ready})_{tmp})$ can be done in $\mathcal{O}(\mid \Pi_i(t_{ready})_{tmp} \mid)$ time.

The following theorem implies that the labeling process of $\pi_i(t)$ can be done in a decreasing order of time.

Theorem 5.1 *The labeling process of $\pi_i(t_{ready})$ depends on the label at time $t^{'} > t_{ready}$ but does not depend on the labels at the lower time period.*

Proof:
Since $\lambda_{ij}(t_{depart}) > 0$, (10) includes only $\pi_j(t_{arrival})$ for $t_{arrival} > t_{depart}$. ■

Corollary 5.1 *The labeling process can be done in a decreasing order of time, i.e. starting from time $t_{ready} = T$ down to 0.*

This corollary implies that the algorithm is started by processing the set $\Pi_i(t_{ready}-1)_{tmp}$ for any $i \in N - \{d\}$ only when $\Pi_j(t_{ready})_{tmp} = \emptyset$ for all $j \in N$ and $t_{ready} \in \{1, \dots, T\}$. Consequently, the selection of a pivot label $\pi_j^*(t_{arrival}) \in \Pi_j(t_{arrival})_{tmp}$ can be done arbitrarily, as stated by the following proposition.

Proposition 5.1 *An arbitrary selection rule can be applied to choose a pivot label $\pi_j^*(t_{arrival})$ from the set $\Pi_j(t_{arrival})_{tmp}$. In effect, each member of $\Pi_j(t)_{prm}$ for any $t \in \{0, \dots, T\}$ defines a Pareto optimal $j - d$ dynamic path with ready time t.*

Proof:
The merging process (11) implies that $\Pi_j(t)_{tmp}$ contains only nonequivalent and Pareto optimal labels. Furthermore, Corollary 5.1 implies that during the labeling process at time t, no additional label is inserted into or deleted from $\Pi_j(t)_{tmp}$. The members of $\Pi_j(t)_{tmp}$ are moved to $\Pi_j(t)_{prm}$ one by one and the labeling process at time t finishes when $\Pi_j(t)_{tmp}$ is empty. Therefore, an arbitrary selection rule for the pivot label can be applied and guarantees that $\Pi_j(t)_{prm}$ contains only nonequivalent and Pareto optimal labels. ■

This proposition implies that $\pi_j^*(t_{arrival})$ can be chosen in constant time $\mathcal{O}(1)$ by e.g. maintaining the set $\Pi_j(t_{arrival})_{tmp}$ as a linked list and take $\pi_j^*(t_{arrival})$ from the head of this list.

In order to do labeling in a decreasing order of time, we need to determine

- $Arrival_{ij}^{-1}(t_{arrival})$, the set of all possible departure times in $\{0, \dots, T\}$ along arc $(i, j) \in A$ arriving at node j at time $t_{arrival}$, i.e.

$$Arrival_{ij}^{-1}(t_{arrival}) \quad := \quad \{t^{'} \ : \ t^{'} + \lambda_{ij}(t^{'}) = t_{arrival}\} \qquad (12)$$

- $Depart_i^{-1}(t_{depart})$, the set of all possible ready times in $\{0, \dots, T\}$ at node $i \in N - \{d\}$ departing from node i at time t_{depart}, i.e.

$$\begin{aligned} Depart_i^{-1}(t_{depart}) \quad := \quad & \{t^{'} \ : \ 0 \le t \le t^{'} \le t_{depart} \text{ with} \\ & \prod_{t^{''}=t}^{t_{depart}-1} w_i(t^{''}) > 0 \text{ and } w_i(t-1) = 0\} \end{aligned} \qquad (13)$$

The set $Arrival_{ij}^{-1}(t_{arrival})$ may be empty. But, the set $Depart_i^{-1}(t_{depart})$ has cardinality at least one, since $t_{depart} \in \{0, \dots, T\}$ itself must be in this set.

These two inverse sets can be determined in a preprocessing step before the labeling process. The alternative is to compute these inverse sets as needed while employing the labeling algorithm.

The complete algorithm to find the minimal complete set of all PO-paths from every node $i \in N - \{d\}$ in the network to the sink d and for all ready times $t \in \{0, \dots, T\}$ is given in Algorithm 5.2. At the end of algorithm there are as many permanent labels associated with node $i \in N$ and ready time $t \in \{0, \dots, T\}$ as the number of nonequivalent Pareto optimal dynamic paths from i with ready time t to the sink node. The correctness of Algorithm 5.2 is shown by the following proposition.

Algorithm 5.2

INPUT Network $G = (N, A, T)$, $w(t), c(t)$, and $h(t)$.
OUTPUT $PO(\mathbb{P}_{id}(t))$, $\forall i \in N$, $\forall t \in \{0, \dots, T\}$.

0 Modify the cost vector $c_{ij}(t)$, $\forall t \in \{0, \dots, T\}$ as follows.

$$c_{ij}(t) = \begin{cases} \begin{pmatrix} c^1_{ij}(t) \\ c^2_{ij}(t) \end{pmatrix} & , t + \lambda_{ij}(t) \leq T \\ \begin{pmatrix} \infty \\ \infty \end{pmatrix} & , t + \lambda_{ij}(t) > T \end{cases} \quad ; \forall (i,j) \in A$$

1 Define the initial set of temporary labels to every node $i \in N$ and for all $t \in \{0, \dots, T\}$ as

$$\Pi_i(t)_{tmp} := \begin{cases} \left\{ \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} \; ; \; (t,t) \; ; \; \emptyset \right) \right\} & , i = d \\ \emptyset & , \text{ otherwise} \end{cases}$$

Assign the empty set as the initial set of permanent labels to every node $i \in N$ and for all $t \in \{0, \dots, T\}$, i.e. $\Pi_i(t)_{prm} := \emptyset$.

2 For $t_{arrival} = T$ down to 0 do
{
  While $\bigcup_{i \in N} \Pi_i(t_{arrival})_{tmp} \neq \emptyset$ do
  { Select a pivot label $\pi^*_j(t_{arrival})$ from $\Pi_j(t_{arrival})_{tmp}$
    Remove $\pi^*_j(t_{arrival})$ from $\Pi_j(t_{arrival})_{tmp}$ to $\Pi_j(t_{arrival})_{prm}$.
    For all $(i,j) \in A$ do
    { For each $t_{depart} \in Arrival^{-1}_{ij}(t_{arrival})$ do
      {For each $t_{ready} \in Depart^{-1}_i(t_{depart})$ do
        { $\hat{\pi}_i(t_{ready}) := \pi^*_j(t_{arrival}) + \big(c_{ij}(t_{depart}) \oplus H_i(t_{ready}, t_{depart})\big)$
          $\Pi_i(t_{ready})_{tmp} := Merge\big(\Pi_i(t_{ready})_{tmp}, \{\hat{\pi}_i(t_{ready})\}\big)$
        }
      }
    }
  }
}

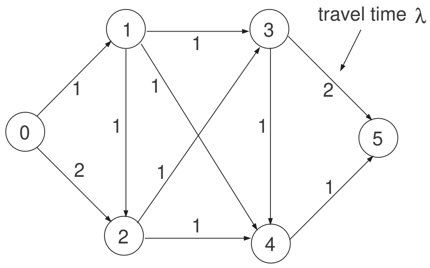
3 For any $i \in N$ and any $t \in \{0, \dots, T\}$, the set $PO(\mathbb{P}_{id}(t))$ is obtained by forwardtracking the successor pointers of $\Pi_i(t)_{prm}$.
Terminate the algorithm.

Proposition 5.2 *Algorithm 5.2 generates $PO(\mathbb{P}_{id}(t))$ for any node $i \in N$ and ready time $t \in \{0, \dots, T\}$.*

Proof:
By (11), $\Pi_i(t)_{tmp}$ contains only nonequivalent and Pareto optimal labels. This set $\Pi_i(t)_{tmp}$ will be emptied and its content is moved into $\Pi_i(t)_{prm}$ during the labeling process at time t. Theorem 5.1 and Proposition 5.1 imply that $\Pi_i(t)_{prm}$ is a minimal complete set. Furthermore, in step 2, all possible candidates for the Pareto optimal labels of any predecessor nodes $i \in N$ are tested from all permanent Pareto optimal labels of the successor node $j \in N$ with $(i, j) \in A$. This is correct since by Theorem 3.1, no Pareto optimal path can be constructed from a dominated sub-path. ■

Later in Section 7 we compare the performance of Algorithms 5.1 and 5.2. The following example illustrates Algorithm 5.2.

Example 5.1
Consider the network in Figure 6. Node 5 is the sink node. We define T equals



$$\lambda_{23}(t) = \begin{cases} 2 & ,t \leq 3 \\ 1 & ,t > 3 \end{cases}$$

Figure 6: Network for Example 5.1

8 time units. The maximum waiting w, cost data corresponding to the travel cost c of two criteria and waiting cost h for each arcs and for any time t in $\{0, \ldots, T\}$ are given below.

$$c_{01}(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \ t \leq 8 \ ; \quad c_{02}(t) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \ t \leq 8 \ ; \ c_{12}(t) = \begin{pmatrix} 7 \\ 8 \end{pmatrix}, \ t \leq 8$$

$$c_{13}(t) = \begin{cases} \begin{pmatrix} 2 \\ 6 \end{pmatrix} & ,t \leq 3 \\ \begin{pmatrix} 4 \\ 4 \end{pmatrix} & ,t > 3 \end{cases} \ ; \ c_{14}(t) = \begin{cases} \begin{pmatrix} 3 \\ 1 \end{pmatrix} & ,t \leq 3 \\ \begin{pmatrix} 5 \\ 3 \end{pmatrix} & ,t > 3 \end{cases} \ ;$$

$$c_{23}(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \ t \leq 8 \ ; \ c_{24}(t) = \begin{pmatrix} 1 \\ 5 \end{pmatrix}, \ t \leq 8 \ ; \ c_{34}(t) = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \ \forall t \leq 8 \ ;$$

$$c_{35}(t) = \begin{cases} \begin{pmatrix} 2 \\ 2 \end{pmatrix} & ,t \leq 2 \\ \begin{pmatrix} 4 \\ 3 \end{pmatrix} & ,t > 2 \end{cases} \ ; \ c_{45}(t) = \begin{cases} \begin{pmatrix} 4 \\ 3 \end{pmatrix} & ,t \leq 3 \\ \begin{pmatrix} 0 \\ 2 \end{pmatrix} & ,t > 3 \end{cases}$$

$$H_i(t, t') = (t' - t), \ \forall t' \geq t \ ; \ w_i(t) = \begin{cases} 1 & ,i = 2, 3, 4 \text{ and } t \geq 4 \\ 0 & ,\text{otherwise} \end{cases}$$

The final label $\Pi_i(t)_{prm}$ for every node $i \in N$ and for every time $t \in \{0, \dots, T\}$ are reported in Table 2 and Table 3.

Node i	$\Pi_i(8)_{prm}$	$\Pi_i(7)_{prm}$	$\Pi_i(6)_{prm}$
5	$\left\{\left(\begin{pmatrix}0\\0\end{pmatrix};(8,8);\emptyset\right)\right\}$	$\left\{\left(\begin{pmatrix}0\\0\end{pmatrix};(7,7);\emptyset\right)\right\}$	$\left\{\left(\begin{pmatrix}0\\0\end{pmatrix};(6,6);\emptyset\right)\right\}$
4	$\emptyset$	$\left\{\left(\begin{pmatrix}0\\2\end{pmatrix};(7,7);5^1(8)\right)\right\}$	$\left\{\left(\begin{pmatrix}0\\2\end{pmatrix};(6,6);5^1(7)\right)\right\}$
3	$\emptyset$	$\emptyset$	$\left\{\left(\begin{pmatrix}4\\3\end{pmatrix};(6,6);5^1(8)\right), \left(\begin{pmatrix}2\\4\end{pmatrix};(6,6);4^1(7)\right)\right\}$
2	$\emptyset$	$\emptyset$	$\left\{\left(\begin{pmatrix}1\\7\end{pmatrix};(6,6);4^1(7)\right)\right\}$
1	$\emptyset$	$\emptyset$	$\left\{\left(\begin{pmatrix}5\\5\end{pmatrix};(6,6);4^1(7)\right)\right\}$
0	$\emptyset$	$\emptyset$	$\emptyset$

Node i	$\Pi_i(5)_{prm}$	$\Pi_i(4)_{prm}$	$\Pi_i(3)_{prm}$
5	$\left\{\left(\begin{pmatrix}0\\0\end{pmatrix};(5,5);\emptyset\right)\right\}$	$\left\{\left(\begin{pmatrix}0\\0\end{pmatrix};(4,4);\emptyset\right)\right\}$	$\left\{\left(\begin{pmatrix}0\\0\end{pmatrix};(3,3);\emptyset\right)\right\}$
4	$\left\{\left(\begin{pmatrix}0\\2\end{pmatrix};(5,5);5^1(6)\right)\right\}$	$\left\{\left(\begin{pmatrix}0\\2\end{pmatrix};(4,4);5^1(5)\right)\right\}$	$\left\{\left(\begin{pmatrix}4\\3\end{pmatrix};(3,3);5^1(4)\right)\right\}$
3	$\left\{\left(\begin{pmatrix}4\\3\end{pmatrix};(5,5);5^1(6)\right), \left(\begin{pmatrix}2\\4\end{pmatrix};(5,5);4^1(6)\right)\right\}$	$\left\{\left(\begin{pmatrix}4\\3\end{pmatrix};(4,4);5^1(5)\right), \left(\begin{pmatrix}2\\4\end{pmatrix};(4,4);4^1(5)\right)\right\}$	$\left\{\left(\begin{pmatrix}4\\3\end{pmatrix};(3,3);5^1(4)\right), \left(\begin{pmatrix}2\\4\end{pmatrix};(3,3);4^1(4)\right)\right\}$
2	$\left\{\left(\begin{pmatrix}5\\4\end{pmatrix};(5,5);3^1(6)\right), \left(\begin{pmatrix}3\\5\end{pmatrix};(5,5);3^2(6)\right), \left(\begin{pmatrix}1\\7\end{pmatrix};(5,5);4^1(6)\right)\right\}$	$\left\{\left(\begin{pmatrix}5\\4\end{pmatrix};(4,4);3^1(5)\right), \left(\begin{pmatrix}3\\5\end{pmatrix};(4,4);3^2(5)\right), \left(\begin{pmatrix}1\\7\end{pmatrix};(4,4);4^1(5)\right)\right\}$	$\left\{\left(\begin{pmatrix}5\\4\end{pmatrix};(3,3);3^1(5)\right), \left(\begin{pmatrix}3\\5\end{pmatrix};(3,3);3^2(4)\right), \left(\begin{pmatrix}1\\7\end{pmatrix};(3,3);4^1(4)\right)\right\}$
1	$\left\{\left(\begin{pmatrix}5\\5\end{pmatrix};(5,5);4^1(6)\right)\right\}$	$\left\{\left(\begin{pmatrix}5\\5\end{pmatrix};(4,4);4^1(5)\right)\right\}$	$\left\{\left(\begin{pmatrix}3\\3\end{pmatrix};(3,3);4^1(4)\right)\right\}$
0	$\left\{\left(\begin{pmatrix}6\\6\end{pmatrix};(5,5);1^1(6)\right)\right\}$	$\left\{\left(\begin{pmatrix}6\\6\end{pmatrix};(4,4);1^1(5)\right), \left(\begin{pmatrix}3\\8\end{pmatrix};(4,4);2^1(6)\right)\right\}$	$\left\{\left(\begin{pmatrix}7\\5\end{pmatrix};(3,3);2^1(5)\right), \left(\begin{pmatrix}5\\6\end{pmatrix};(3,3);2^2(5)\right), \left(\begin{pmatrix}3\\8\end{pmatrix};(3,3);2^3(5)\right)\right\}$

Table 2: Labels $\Pi_i(t)_{prm}$ for every node $i \in N$ and for any time $t \in \{0, \dots, T\}$ of the network Example 5.1

The Pareto optimal dynamic paths from node 0 to node 5 that are ready to

Node i	$\Pi_i(2)_{prm}$	$\Pi_i(1)_{prm}$	$\Pi_i(0)_{prm}$
5	$\left\{\left(\begin{pmatrix}0\\0\end{pmatrix};(2,2);\emptyset\right)\right\}$	$\left\{\left(\begin{pmatrix}0\\0\end{pmatrix};(1,1);\emptyset\right)\right\}$	$\left\{\left(\begin{pmatrix}0\\0\end{pmatrix};(0,0);\emptyset\right)\right\}$
4	$\left\{\left(\begin{pmatrix}4\\3\end{pmatrix};(2,2);5^1(3)\right)\right\}$	$\left\{\left(\begin{pmatrix}4\\3\end{pmatrix};(1,1);5^1(2)\right)\right\}$	$\left\{\left(\begin{pmatrix}4\\3\end{pmatrix};(0,0);5^1(1)\right)\right\}$
3	$\left\{\left(\begin{pmatrix}2\\2\end{pmatrix};(2,2);5^1(4)\right)\right\}$	$\left\{\left(\begin{pmatrix}2\\2\end{pmatrix};(1,1);5^1(3)\right)\right\}$	$\left\{\left(\begin{pmatrix}2\\2\end{pmatrix};(0,0);5^1(2)\right)\right\}$
2	$\left\{\left(\begin{pmatrix}5\\4\end{pmatrix};(2,2);3^1(4)\right), \left(\begin{pmatrix}3\\5\end{pmatrix};(2,2);3^2(4)\right)\right\}$	$\left\{\left(\begin{pmatrix}5\\4\end{pmatrix};(1,1);3^1(3)\right), \left(\begin{pmatrix}3\\5\end{pmatrix};(1,1);3^2(3)\right)\right\}$	$\left\{\left(\begin{pmatrix}3\\3\end{pmatrix};(0,0);3^1(2)\right)\right\}$
1	$\left\{\left(\begin{pmatrix}7\\4\end{pmatrix};(2,2);4^1(3)\right), \left(\begin{pmatrix}6\\9\end{pmatrix};(2,2);3^1(3)\right), \left(\begin{pmatrix}4\\10\end{pmatrix};(2,2);3^2(3)\right)\right\}$	$\left\{\left(\begin{pmatrix}7\\4\end{pmatrix};(1,1);4^1(2)\right), \left(\begin{pmatrix}4\\8\end{pmatrix};(1,1);3^1(2)\right)\right\}$	$\left\{\left(\begin{pmatrix}7\\4\end{pmatrix};(0,0);4^1(1)\right), \left(\begin{pmatrix}4\\8\end{pmatrix};(0,0);3^1(1)\right)\right\}$
0	$\left\{\left(\begin{pmatrix}4\\4\end{pmatrix};(2,2);1^1(3)\right), \left(\begin{pmatrix}3\\8\end{pmatrix};(2,2);2^3(4)\right)\right\}$	$\left\{\left(\begin{pmatrix}7\\5\end{pmatrix};(1,1);2^1(3)\right), \left(\begin{pmatrix}5\\6\end{pmatrix};(1,1);2^2(3)\right), \left(\begin{pmatrix}3\\8\end{pmatrix};(1,1);2^3(3)\right)\right\}$	$\left\{\left(\begin{pmatrix}7\\5\end{pmatrix};(0,0);2^1(2)\right), \left(\begin{pmatrix}5\\6\end{pmatrix};(0,0);2^2(2)\right)\right\}$

Table 3: (Cont.) Labels $\Pi_i(t)_{prm}$ for every node $i \in N$ and for any time $t \in \{0, \dots, T\}$ of the network Example 5.1

leave node 0 at time $t = 0$ are

$$\{0(0,0), 2(2,2), 3(4,4), 5(6,6)\} \quad \text{with value} \quad \begin{pmatrix}7\\5\end{pmatrix}$$

$$\{0(0,0), 2(2,2), 3(3,3), 4(4,4), 5(5,5)\} \quad \text{with value} \quad \begin{pmatrix}5\\6\end{pmatrix}$$

Hence,

$$\begin{aligned} PO(\mathbb{P}_{05}(0)) &= \{\{0(0,0), 2(2,2), 3(4,4), 5(6,6)\}, \{0(0,0), 2(2,2), \\ &\qquad 3(3,3), 4(4,4), 5(5,5)\}\} \end{aligned}$$

We see from Table 3 that there is no path which can leave node 0 after time $t = 5$ and arrive at node 5 in time horizon $T = 8$ or earlier. $\square$

6 Algorithm for All Ready Times with Arbitrary Data

When negative travel times and costs are allowed, Theorem 5.1 is no longer valid. Moreover, as shown in Example 3.3, there may exist some negative dynamic cycles within $\{0, \dots, T\}$. Consequently, dynamic shortest path algorithms must be able to detect negative dynamic cycles.

We describe here an algorithm based on the label-correcting technique. Inside we use a scan eligible (SE)-list to store the nodes which have potential of improving the labels of at least one other node. During the initialization step of the algorithm, only the sink node is in the SE-list. The labeling process moves backward from the sink to all other nodes. The label structure of node i at time t is given by (6) and $\Pi_i(t)$ denotes the set of all cost labels associated with node i of ready time t. At each iteration of the algorithm, a node, called the *current node*, is removed from the SE list. In the first iteration, the sink node is defined as the current node. During the scanning process of the current node, the temporary labels of all predecessor nodes of the current node are calculated for each time step $t \in \{0, \dots, T\}$. As the algorithm progresses, the label $\Pi_i(t)$ is updated in such a way that $\Pi_i(t)$ is always a Pareto optimal set. The cardinality of $\Pi_i(t)$ may increase or decrease as the algorithm proceeds. Upon termination, this set contains the labels denoting all Pareto optimal dynamic paths from node i to the sink node d with ready time t. The current node is selected from the SE-list following the FIFO rule.

Suppose that node i is a predecessor node of the current node j. We denote by $\{tmpLabel\}$ the set of temporary labels of node i at time $t_{ready} \in \{0, \dots, T\}$ generated by j. Recall the definition of $Depart_i(t_{ready})$, the set of all possible departure times in $\{0, \dots, T\}$ of the ready time t_{ready}, given by (7). Using $Depart_i(t_{ready})$, we define

$$\begin{aligned} \{tmpLabel\} \; := \; & \Pi_j^*(t_{arrival}) + (c_{ij}(t_{depart}) \oplus H_i(t_{ready}, t_{depart})), \\ & \text{for all } t_{depart} \in Depart_i(t_{ready}) \\ & \text{with } t_{arrival} = t_{depart} + \lambda_{ij}(t_{depart}) \end{aligned} \tag{14}$$

The set $\{tmpLabel\}$ is then merged with the current label of node i at time t_{ready}, $\Pi_i(t_{ready})$, yielding $\Pi_i(t_{ready})_{new}$, i.e.

$$\Pi_i(t_{ready})_{new} = Merge(\Pi_i(t_{ready}), \{tmpLabel\}) \tag{15}$$

The definition of Merge function is given by (11). If $\Pi_i(t_{ready})_{new}$ is not equal to $\Pi_i(t_{ready})$, then we replace $\Pi_i(t_{ready})$ by $\Pi_i(t_{ready})_{new}$ and send the node i to the tail of SE list, i.e. $SE := SE + \{i\}$. We then select again a node from the head of the SE-list and repeat the same process until the list SE becomes empty (i.e. no current node can be selected), or the algorithm detects the existence of a negative dynamic cycle.

Lemma 6.1 *If at any time during the labeling algorithm there is a dynamic cycle, then this dynamic cycle is negative.*

Proof:
Suppose the labeling process produces a dynamic path P given by

$$P := \{j_1(t_1, t_1^{'}), j_2(t_2, t_2^{'}), \dots, j_{k-1}(t_{k-1}, t_{k-1}^{'})\}$$

with $j_1 = i$ as shown in Figure 7 (take $k = 4$).

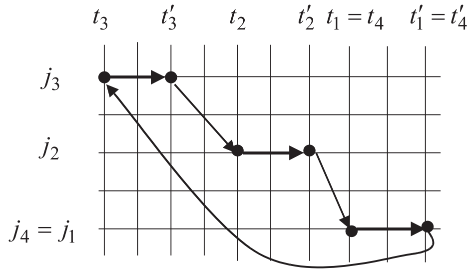


Figure 7: A dynamic cycle for the proof of Lemma 6.1

Since we do cost labeling in a backward direction, the cost label of any node $j_l \in P$ satisfies

$$cost(\pi_{j_{l+1}}(t_{l+1})) = cost(\pi_{j_l}(t_l)) + (c_{j_{l+1},j_l}(t'_{l+1}) \oplus H_{j_{l+1}}(t_{l+1}, t'_{l+1}))$$

By taking the sum over $l = 1$ to $k-2$, we obtain

$$cost(\pi_{j_{k-1}}(t_{k-1})) = cost(\pi_{j_1}(t_1)) + \sum_{l=1}^{k-2}(c_{j_{l+1},j_l}(t'_{l+1}) \oplus H_{j_{l+1}}(t_{l+1}, t'_{l+1})) \quad (16)$$

Now, suppose that the scanning process of the current node j_{k-1} on arc (i, j_{k-1}) creates a dynamic cycle C, i.e. $j_k = j_1 = i$, $t_k = t_1$, $t'_k = t'_1$, and

$$C := \{j_1(t_1, t'_1), j_2(t_2, t'_2), \ldots, j_{k-1}(t_{k-1}, t'_{k-1}), j_k(t_k, t'_k) = j_1(t_1, t'_1)\}$$

This cycle is created only if either $cost(\pi_{j_{k-1}}(t_{k-1})) + (c_{j_1,j_{k-1}}(t'_1) \oplus H_{j_1}(t_1, t'_1))$ dominates $cost(\pi_{j_1}(t_1))$ or both $cost(\pi_{j_{k-1}}(t_{k-1})) + (c_{j_1,j_{k-1}}(t'_1) \oplus H_{j_1}(t_1, t'_1))$ and $cost(\pi_{j_1}(t_1))$ are Pareto optimal. If this scanning process replaces the current label of $\pi_{j_1}(t_1)$ (i.e. $cost(\pi_{j_{k-1}}(t_{k-1})) + (c_{j_1,j_{k-1}}(t'_1) \oplus H_{j_1}(t_1, t'_1))$ dominates $cost(\pi_{j_1}(t_1))$), then just before the scanning process we must have

$$cost(\pi_{j_1}(t_1)) > cost(\pi_{j_{k-1}}(t_{k-1})) + (c_{j_1,j_{k-1}}(t'_1) \oplus H_{j_1}(t_1, t'_1)) \quad (17)$$

The sum of (16) and (17) yields

$$\begin{aligned} cost(\pi_{j_{k-1}}(t_{k-1})) + cost(\pi_{j_1}(t_1)) &> cost(\pi_{j_1}(t_1)) + cost(\pi_{j_{k-1}}(t_{k-1})) + \\ &\quad \sum_{l=1}^{k-1}(c_{j_{l+1},j_l}(t'_{l+1}) \oplus H_{j_{l+1}}(t_{l+1}, t'_{l+1})) \\ 0 &> \mathbf{c}(C) \end{aligned} \quad (18)$$

On the other hand, if both $cost(\pi_{j_{k-1}}(t_{k-1})) + (c_{j_1,j_{k-1}}(t'_1) \oplus H_{j_1}(t_1, t'_1))$ and $cost(\pi_{j_1}(t_l))$ are Pareto optimal, then one component of

$$cost(\pi_{j_{k-1}}(t_{k-1})) + (c_{j_1,j_{k-1}}(t'_1) \oplus H_{j_1}(t_1, t'_1))$$

must be smaller than the corresponding component of $cost(\pi_{j_1}(t_l))$ and the other component is bigger than that of $cost(\pi_{j_1}(t_l))$. W.l.o.g. we assume that the first component of $cost(\pi_{j_{k-1}}(t_{k-1})) + (c_{j_1,j_{k-1}}(t_1^{'}) \oplus H_{j_1}(t_1, t_1^{'}))$ is smaller than the first component of $cost(\pi_{j_1}(t_l))$, i.e.

$$cost^1(\pi_{j_1}(t_l)) > cost^1(\pi_{j_{k-1}}(t_{k-1})) + (c^1_{j_1,j_{k-1}}(t_1^{'}) + H_{j_1}(t_1, t_1^{'})) \tag{19}$$

The sum of the first component of (16) and (19) yields

$$0 > \mathbf{c}^1(C) \tag{20}$$

Hence, C is a negative dynamic cycle. ■

Proposition 6.1 *Suppose that for some node $i \in N - \{d\}$ and some time $t \in \{0, \ldots, T\}$, there is some dynamic $i - d$ path with ready time t containing a negative dynamic cycle.*

(a) *If both components of the cost of the negative dynamic cycle are negative, then $PO(\mathbb{P}_{id}(t)) = \emptyset$.*

(b) *If one component of the cost of the negative dynamic cycle is negative and the other is zero, then $PO(\mathbb{P}_{id}(t)) = \emptyset$.*

(c) *If one component of the negative dynamic cycle is negative and the other is positive, then there will be an infinite number of Pareto optimal dynamic paths.*

Proof:
Let C be a negative dynamic cycle associated with the label $\pi_i(t)$.

(a) Suppose that $cost^1(\pi_i(t)) < 0$ and $cost^2(\pi_i(t)) < 0$. Let's construct a new label of node i at time t, $\pi_i^{'}(t)$, obtained by repeating the cycle C two times. We obtain

$$cost^1(\pi_i^{'}(t)) = 2cost^1(\pi_i(t)) < 0 \quad \text{and} \quad cost^2(\pi_i^{'}(t)) = 2cost^2(\pi_i(t)) < 0$$

Obviously, $\pi_i^{'}(t)$ dominates $\pi_i(t)$. Since we can repeat the cycle infinitely, the labeling process will always be able to produce a new $\pi_i(t)$ that dominates the current label $\pi_i(t)$. Hence, there will be no Pareto optimal $\pi_i(t)$.

(b) Wlog, assume that $cost^1(\pi_i(t)) < 0$ and $cost^2(\pi_i(t)) = 0$. As in the proof of (a), we can have label $\pi_i^{'}(t)$ such that

$$cost^1(\pi_i^{'}(t)) = 2cost^1(\pi_i(t)) < 0 \quad \text{and} \quad cost^2(\pi_i^{'}(t)) = 2cost^2(\pi_i(t)) = 0$$

Therefore, $\pi_i^{'}(t)$ dominates $\pi_i(t)$. Since we can repeat the cycle infinitely, there will be no Pareto optimal $\pi_i(t)$.

(c) Wlog, assume that $cost^1(\pi_i(t)) > 0$ and $cost^2(\pi_i(t)) < 0$. As in the proof of (a), let's construct a new label $\pi_i^{'}(t)$ by repeating C two times. We obtain

$$cost^1(\pi_i^{'}(t)) = 2cost^1(\pi_i(t)) > 0 \quad \text{and} \quad cost^2(\pi_i^{'}(t)) = 2cost^2(\pi_i(t)) < 0$$

where

$$cost^1(\pi_i^{'}(t)) > 2cost^1(\pi_i(t)) \quad \text{and} \quad cost^2(\pi_i^{'}(t)) < 2cost^2(\pi_i(t))$$

Therefore, both $\pi_i^{'}(t)$ and $\pi_i(t)$ are Pareto optimal paths. Since we can repeat this cycle infinitely, there will be an infinite number of Pareto optimal dynamic paths. ■

To detect the existence of a negative dynamic cycle, we divide the execution of the algorithm into *passes*. The definition of passes is as given as folows.

Definition 6.1

- *Pass* 0 *ends after node* d *is scanned for the first time.*
- *Pass* k *ends after all nodes in the* SE*-list at the end of pass* $k-1$ *have been scanned.*

From this definition, if a node i is removed from the list before the end of pass k, then there must be a node j with $(i,j) \in A$ removed from the list before the end of pass $(k-1)$ and there are $t^{'}, t \in \{0, \ldots, T\}$ such that $\Pi_i(t)$ is improved from $\Pi_j(t^{'})$. We denote by ps the index of the pass of the execution of the algorithm. From the definition of a pass, we increase ps by one if at the end of pass ps, the list SE is not empty. We denote by ls the node in the tail of SE-list at the end of pass $k-1$, i.e.

$$ls := tail(SE)$$

If the current node is equal to ls at the end of pass k and $SE \neq \emptyset$, then

$$ps := ps + 1$$

Lemma 6.2 *If the SE-list is not empty at the end of pass* $(n-1)(T+1)$*, then the dynamic network must contain a negative dynamic cycle.*

Proof:
Define ps_i to be the greatest k such that node i is selected as the current node during pass k. If ps_i is positive, then there must be a node j with $(i,j) \in A$ and some $t, t^{'}$ such that $\Pi_i(t)$ is updated from $\Pi_j(t^{'})$ during pass $(ps_i - 1)$, i.e. node j at time $t^{'}$ becomes a successor component of some member of $\Pi_i(t)$. Since the label of node j may be updated again at a pass greater than ps_i, we obtain the following interrelation

$$ps_i \leq ps_j + 1$$

The value of ps_j is strictly greater than $(ps_i - 1)$ only if the label of node j is updated again before the end of pass greater or equal than ps_i. Therefore, the ps values of any two consecutive nodes in a dynamic path differs by one. Since it is assumed that no arc leaves node d, we obtain $ps_d = 0$. Furthermore, we know that the maximum number of nodes but the sink node in the time-expanded network for $\{0, \ldots, T\}$ is $(n-1)(T+1)$. Now, suppose the algorithm runs until a node i is selected as the current node in pass $(n-1)(T+1)+1$. Since $ps_i = (n-1)(T+1)+1$, then we must eventually repeat some nodes j at time $t^{'} \in \{0, \ldots, T\}$ during the forwardtracking process to find an $i-d$ Pareto optimal dynamic path that is ready at node i at time $t \in \{0, \ldots, T\}$. Hence, this path contains a dynamic cycle. By Theorem 6.1, this cycle must be a negative dynamic cycle. ■

Hence, the existence of a negative cycle can be detected by either finding a dynamic cycle (see Lemma 6.1) or checking the number of passes.

Since $\{tmpLabel\}$ is in general not a singleton, the merging process in (15) may not be as simply as that in Section 5. Brumbaugh-Smith and Shier [4] proposed an algorithm for a merge operation of two Pareto optimal sets, B_1 and B_2, which has a linear worst-case time complexity of $\mathcal{O}(\mid B_1 \mid + \mid B_2 \mid)$. This method works on assumption that B_1 and B_2 are ordered sets with respect to their cost components, arranged by increasing value of the first criteria of the cost component. Hence, if the cardinality of B_1 is k_1 and $cost(B_1)$ defines the set of cost components of all members of B_1, then

$$\begin{aligned} cost(B_1) \quad = \quad & \left\{ \begin{pmatrix} u_1^1 \\ u_1^2 \end{pmatrix}, \begin{pmatrix} u_2^1 \\ u_2^2 \end{pmatrix}, \ldots, \begin{pmatrix} u_{k_1}^1 \\ u_{k_1}^2 \end{pmatrix} \; : \; l^{'} < l^{''} \Rightarrow u_{l^{'}}^1 < u_{l^{''}}^1, \right. \\ & \left. \qquad\qquad\qquad\qquad\qquad l^{'}, l^{''} \in \{1, \ldots, k_1\} \right\} \end{aligned}$$

Since $cost(B_1)$ contains only different Pareto optimal values, it follows that the values corresponding to the second criteria form a strictly decreasing sequence, i.e. $l^{'} < l^{''} \Rightarrow u_{l^{'}}^2 > u_{l^{''}}^2$.

Let us denote by $time(u)$ and $point(u)$ the time component and the pointer component of $u \in B_1$, respectively. Then B_1 can be represented as

$$\begin{aligned} B_1 \quad = \quad & \Big\{ \Big(cost(u_1); time(u_1); point(u_1) \Big), \ldots, \\ & \quad \Big(cost(u_{k_1}); time(u_{k_1}); point(u_{k_1}) \Big) \; : \; l^{'} < l^{''} \Rightarrow u_{l^{'}}^1 < u_{l^{''}}^1; \\ & \qquad\qquad\qquad\qquad\qquad l^{'}, l^{''} \in \{1, \ldots, k_1\} \Big\} \end{aligned}$$

The same arrangement is applied to B_2, i.e. if k_2 is the cardinality of B_2, then

$$\begin{aligned} B_2 \quad = \quad & \Big\{ \Big(cost(v_1); time(v_1); point(v_1) \Big), \ldots, \\ & \quad \Big(cost(v_{k_2}); time(v_{k_2}); point(v_{k_2}) \Big) \; : \; l^{'} < l^{''} \Rightarrow v_{l^{'}}^1 < v_{l^{''}}^1; \\ & \qquad\qquad\qquad\qquad\qquad l^{'}, l^{''} \in \{1, \ldots, k_2\} \Big\} \end{aligned}$$

We define *the set concatenation operation* "&" as

$$\{u_1, \ldots, u_{k_1}\} \& \{v_1, \ldots, v_{k_2}\} \quad := \quad \{u_1, \ldots, u_{k_1}, v_1, \ldots, v_{k_2}\}$$

Details of the modified Merge procedure are given in Algorithm 6.1.

Algorithm 6.1 (adapted from [4])

INPUT	B_1 and B_2 are ordered sets arranged by increasing value of the first criteria of the cost component.
OUTPUT	$Merge(B_1, B_2)$
0	Initiate counter $i = 1$ and $j = 1$. Initiate Merge set $M = \emptyset$.

1	If $i > k_1$ then
	    $M = M\&\{v_j, \dots, v_{k_2}\}$
	    Go to Step 3.
	If $j > k_2$ then
	    $M = M\&\{u_i, \dots, u_{k_1}\}$
	    Go to Step 3.

2	If $u_i^1 < v_j^1$ then
	    while $u_i^2 \leq v_j^2$
	        $j = j + 1$
	    $M = M\&\{u_i\}$
	    $i = i + 1$ and Go to Step 1.
	Else if $v_j^1 < u_i^1$ then
	    while $v_j^2 \leq u_i^2$
	        $i = i + 1$
	    $M = M\&\{v_j\}$
	    $j = j + 1$ and Go to Step 1.
	Else if $u_i^2 < v_j^2$ then
	    repeat
	        $j = j + 1$
	    until $v_j^2 < u_i^2$
	    $M = M\&\{u_i\}$
	    $i = i + 1$ and Go to Step 1.
	Else
	    repeat
	        $i = i + 1$
	    until $u_i^2 < v_j^2$
	    $M = M\&\{v_j\}$
	    $j = j + 1$ and Go to Step 1.

3	$Merge(B_1, B_2) := M$ Terminate the algorithm.

Proposition 6.2 *Algorithm* 6.1 *produces a minimal complete set M of* $B_1 \cup B_2$ *in* $\mathcal{O}(\mid B_1 \mid + \mid B_2 \mid)$.

Proof:
The proof focusses on step 2 of the algorithm. For any $i \leq k_1$ and $j \leq k_2$, if $u_i^1 < v_j^1$ then we check the value of u_i^2 compared with v_j^2. If $u_i^2 \leq v_j^2$, then u_i dominates v_j. Therefore, we skip v_j and remove it from further consideration as a member of M by increasing the counter j. Since v_j^1 is increasing and v_j^2 is decreasing, we can continue removing v_j by increasing the counter j as long as $u_i^2 \leq v_j^2$. Furthermore, since u_i^1 and v_j^1 are increasing, u_i is Pareto optimal and placed into the set M. We then increase the counter i and check again

the possibility to include the new u_i into M by going back to step 1. We do similarly in the case of $v_j^1 < u_i^1$, but for possibly removing u_i from further consideration as a member of M. Moreover, if $u_i^1 = v_j^1$, then we check the value of u_i^2 compared with v_j^2. If $u_i^2 < v_j^2$, then u_i dominates v_j. Since v_i^1 is increasing and v_j^2 is decreasing in B_2, we skip v_j sequentially until we find $v_j^2 < u_i^2$. At this time we can check again the possibility to include v in M. But, if $u_i^2 > v_j^2$, then similarly we skip u_i sequentially until we find $u_i^2 < v_j^2$. In case the two cost values u_i and v_j are identical, i.e. $u_i^1 = v_j^1$ and $u_i^2 = v_j^2$, we disregard u_i keeping the minimal property of the set M. The complexity follows directly by computing the steps of the algorithm. ■

The proposed label correcting algorithm to generate all Pareto optimal $i-d$ paths for any $i \in N$ and for any ready time $t \in \{0, \ldots, T\}$ is shown in Algorithm 6.2.

Theorem 6.1 (Correctness of the Algorithm 6.2) *Algorithm* 6.2 *generates all Pareto optimal paths from every node in the network to the sink and for all ready times $t \in \{0, \ldots, T\}$ in a finite number of steps or it detects a negative dynamic cycle.*

Proof:
If a dynamic cycle is found or $ps = (n-1)(T+1)+1$, then by Lemmas 6.1 and 6.2, the algorithm detects the existence of a negative dynamic cycle. On the other hand, if this is not the case, then the contrapositive of Lemma 6.2 guarantees that Algorithm 6.2 terminates (because the SE-list is empty) at most $(n-1)(T+1)$ passes. Upon termination, for all $i \in N$ and $t \in \{0, \ldots, T\}$, every $\pi_i(t) \in \Pi_i(t))$ defines a Pareto optimal path. Suppose $\exists\ \hat{\pi}_i(t_{ready}) \in \Pi_i(t_{ready})$ with

$$\big(c_{ij}(t_{depart}) \oplus H_i(t_{ready}, t_{depart})\big) + \hat{\pi}_j(t_{arrival}) < \hat{\pi}_i(t_{ready})$$

for some $\hat{\pi}_j(t_{arrival}) \in \big\{\pi_j(t_{arrival}\big\}$.
Since the scanning process in step 3 is done for all $(i,j) \in A$, for all $t_{ready} \in \{0, \ldots, T\}$, and for all $t_{ready} \in Depart_i(t_{ready})$, the condition above implies that node j was not completely scanned yet and must still be in the SE-list, contradicting the termination of the algorithm. ■

The following example illustrates Algorithm 5.2.

Example 6.1
Consider the problem in Example 5.1 (Figure 6), where we only change the data of arc $(3,4)$ to $\lambda_{34} := -1$ and $c_{34}(t) = \begin{pmatrix} 2 \\ -2 \end{pmatrix}$, $t \leq 8$, respectively.

$$\begin{aligned} PO(\mathbb{P}_{05}(0)) \quad = \quad & \{\{0(0,0), 2(2,2), 3(4,5), 4(4,4), 5(5,5)\}, \{0(0,0), 1(1,1), \\ & 3(2,2), 5(4,4)\}\} \end{aligned}$$

with the corresponding optimal values $\begin{pmatrix} 6 \\ 3 \end{pmatrix}$ and $\begin{pmatrix} 5 \\ 9 \end{pmatrix}$. □

Algorithm 6.2

INPUT	Network $G = (N, A, T)$, $w(t), c(t)$, and $h(t)$.
OUTPUT	$PO(\mathbb{P}_{id}(t))$, $\forall i \in N$, $\forall t \in \{0, \ldots, T\}$.
0	Modify the cost vector $c_{ij}(t)$, $\forall t \in \{0, \ldots, T\}$ as follows. $$c_{ij}(t) = \begin{cases} \begin{pmatrix} c^1_{ij}(t) \\ c^2_{ij}(t) \end{pmatrix} & , t + \lambda_{ij}(t) \leq T \\ \begin{pmatrix} \infty \\ \infty \end{pmatrix} & , t + \lambda_{ij}(t) > T \end{cases} \quad ; \ \forall (i,j) \in A$$
1	Assign the label to each node for all $t \in \{0, \ldots, T\}$ $$\Pi_i(t) = \begin{cases} \left\{ \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} \ ; \ (t, t); \emptyset \right) \right\} & , i = d \\ \emptyset & , \text{ otherwise} \end{cases}$$ Set $SE := \{d\}$. Set $ps := 0$; $ls := d$.
2	Take the *current node* j from the head of SE and set $SE := SE - \{j\}$.

3	Scan the current node
	For all $(i, j) \in A$ do
	{ For $t_{ready} = 0$ to T do
	{ For all $t_{depart} \in Depart_i(t_{ready})$ do
	{ If $\Pi_j(t_{depart} + \lambda_{ij}(t_{depart})) \neq \emptyset$, then calculate
	$\{tmpLabel\} := \Pi_j(t_{depart} + \lambda_{ij}(t_{depart}) + \big(c_{ij}(t_{depart}) \oplus H_i(t_{ready}, t_{depart})\big)$
	$\Pi_i(t_{ready})_{new} = Merge\big(\Pi_i(t_{ready}), \{tmpLabel\}\big)$
	If $\Pi_i(t_{ready})_{new} \neq \Pi_i(t_{ready})$, then
	Set $\Pi_i(t_{ready}) := \Pi_i(t_{ready})_{new}$
	Set $SE := SE + \{i\}$
	}
	}
	}

4	If $j = ls$ and $SE \neq \emptyset$, then $ps := ps + 1$; $ls := tail(SE)$ If $ps = (n-1)(T+1) + 1$ or a dynamic cycle is found, then there is a negative dynamic cycle. Terminate the algorithm. If $SE = \emptyset$, then go to Step 5. Return to Step 2.
5	For any $i \in N$ and any $t \in \{0, \ldots, T\}$, the set $PO(\mathbb{P}_{id}(t))$ is obtained by forwardtracking the successor pointers of $\Pi_i(t)$. Terminate the algorithm.

7 Computational Results

Experiments are run to compare the performance of the three algorithms (Algorithms 5.1, 5.2, and 6.2) discussed in the previous sections. Toward this goal, a series of experiments is run on the basis of randomly generated dynamic networks. All travel times and costs are non-negative since only one of the three algorithms can deal with negative data. However, separate experiments with negative data are also run to analyze the performance of Algorithm 6.2, since it is able to deal with negative data. For the experiments, Algorithms 5.1, 5.2, and 6.2 are implemented in C++ and run on a PC Pentium III, 500 MHz, and

RAM of 256 MB.

To run the experiments, a dynamic random network generator is developed by extending the idea of NETGEN [12] to include the time-dependent data. The generated networks are connected. The experiments are conducted on random networks with 50, 100, 500, and 1000 nodes and time horizon $T = 100$. For each choice of n nodes, we create networks with indegree and outdegree of each node 2, 4, 6, and 8. It is assumed that the sink node has zero outdegree. This degree setting implies the generated networks have $2n$, $4n$, $6n$, and $8n$ arcs. The minimum travel time is defined as one. The maximum travel time and costs of both criteria are defined as 10 units. The maximum holding cost is defined as 10. For each specific setting of n and m, we tested five random dynamic networks. Therefore, the total number of observations is 80.

The results of the experiments are given in Table 4. Figure 8 constrasts the

n	m/n	Average CPU-time (in seconds)		
		Algorithm 5.1	5.2	6.2
50	2	2.66	1.01	1.05
	4	6.00	1.64	3.21
	6	6.24	2.29	5.67
	8	14.47	3.00	7.92
100	2	7.37	2.24	2.57
	4	14.61	3.65	7.41
	6	24.63	5.18	12.96
	8	36.98	6.90	19.25
500	2	62.76	14.61	17.82
	4	164.99	24.42	57.73
	6	321.29	35.83	107.75
	8	464.51	48.69	152.03
1000	2	251.85	31.41	44.81
	4	756.21	55.99	154.17
	6	1,269.37	83.87	257.73
	8	2,275.02	117.09	394.18

Table 4: Computational test results

performance of Algorithms 5.1, 5.2, and 6.2, with respect to the CPU-time. This figure shows that the label setting algorithm (Algorithm 5.2) is superior to the other two algorithms.

When negative data is allowed, acyclic networks having a topological order are generated. The maximum value of the parameters are defined as in the case of nonnegative data. The minimum travel time and costs of both criteria are defined as the negative value of their corresponding maximum values. To

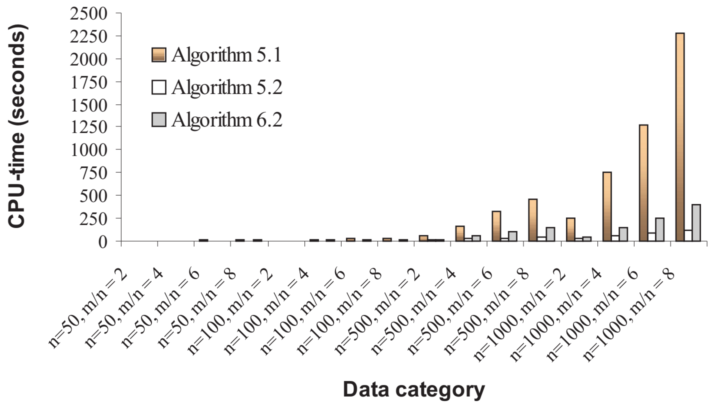


Figure 8: Comparison of the CPU-time of Algorithms 5.1, 5.2, and 6.2

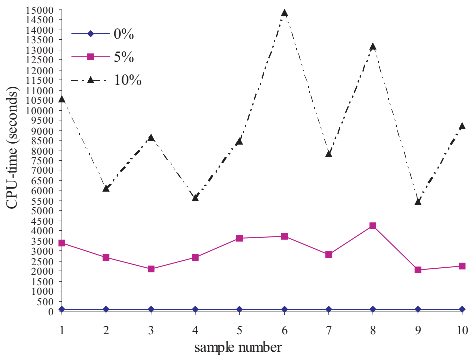


Figure 9: The CPU-time of Algorithm 6.2 for ten random samples with $n = 500$ and $m/n = 8$

control the distribution of data with negative values, we define the percentage of negative data to 0, 5 and 10 percents. The results of experiment with respect to the CPU-time of 10 random networks with size $n = 500$ and density $m/n = 8$ are shown in Table 5. It is found that the *CPU-time increases as the percentage*

percentage of negative data	0%	5%	10%
average of CPU-time (seconds)	104.451	2,956.273	8,995.864
standard deviation	3.726862	757.5691	3,126.029

Table 5: Computational test results of Algorithm 6.2 for three sets of networks with $n = 500$ and $m/n = 8$

of negative parameter increases. Moreover, the *CPU-time tends to be unstable* *as the percentage of negative data increase*, as shown in Figure 9.

8 Conclusions

In this paper, we have presented new label setting and label correcting time-dependent algorithms to solve the bicriterion shortest path problem with time-dependent cost vectors, travel times, and finite waiting at any node but the sink. The label setting algorithm is applicable only when the cost vectors are non-negative in each component and travel times are positive. For general values, a new label correcting algorithm has been introduced. We compared our new algorithm with a dynamic programming-based algorithm due to Kostreva and Wiecek [13]. The computational testing shows that in case of positive valued data, the label setting algorithm is superior to the other two algorithms.

References

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

[2] R. E. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[3] D. Bertesekas. *Linear Network Optimization: Algorithms and Codes.* M.I.T. Press, Cambridge, MA, 1991.

[4] J. Brumbaugh-Smith and D. Shier. An empirical investigation of some bicriterion shortest path algorithms. *European Journal of Operation Research*, 43:216–224, 1989.

[5] I. Chabini. Discrete dynamic shortest path problems in transportation applications. *Transportation Research Record*, 1645:8170–175, 1998.

[6] K. Cooke and E. Halsey. The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis Applications*, 14:493–498, 1966.

[7] H. Corley and I. Moon. Shortest paths in networks with vector weights. *Journal of Optimization Theory and Applications*, 46:79–86, 1985.

[8] S. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17:395–412, 1969.

[9] T. Getachew, M. Kostreva, and L. Lancaster. A generalization of dynamic programming for pareto optimization in dynamic networks. *RAIRO Operation Research*, 34:27–47, 2000.

[10] H. Hamacher and S. Tjandra. Mathematical modeling of evacuation problems: a state of the art. In M. Schreckenberg and S. Sharma, editors, *Pedestrian and Evacuation Dynamics*, pages 227–266, Springer-Verlag, Berlin, 2002.

[11] P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making : Theory and Applications, Lecture Notes in Economics and Mathematical Systems*, volume 177, pages 109–127. Springer-Verlag, Berlin, 1980.

[12] D. Klingman, A. Napier, and J. Stutz. NETGEN : A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20(5):814–821, 1974.

[13] M. Kostreva and M. Wiecek. Time dependency in multiple objective dynamic programming. *Journal of Mathematical Analysis and Application*, 173(1):289–307, 1993.

[14] E. Martins. On a multicriteria shortest path problem. *European Journal of Operation Research*, 16:236–245, 1984.

[15] A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the A.C.M.*, 37(3):607–625, 1990.

[16] A. Orda and R. Rom. Minimum weight paths in time-dependent network. *Networks*, 21(3):295–320, 1991.

[17] S. Pallottino and M. Scutella. Shortest path algorithms in transportation models: classical and innovative aspects. Technical Report 97-06, University of Pisa, Pisa, Italy, 1997.

[18] A. Philpott. Continuous-time shortest path problems and linear programming. *SIAM Journal Control and Optimization*, 32(2):538–552, 1994.

[19] A. Philpott and A. Mees. A finite algorithm for shortest path problems with time-varying costs. *Applied Mathematics Letters*, 6(2):91–94, 1993.

[20] W. Wardell and A. Ziliaskopoulos. A intermodal optimum path algorithm for dynamic multimodal networks. *European Journal of Operation Research*, 125:486–502, 2000.

[21] T. Yamada. A network approach to a city emergency evacuation planning. *International Journal of Systems Science*, 27(10):931–936, 1996.

[22] A. Ziliaskopoulos and H. Mahmassani. Time-dependent shortest path algorithm for real-time intelligent vehicle/highway system. *Transportation Research Record*, 1408:94–104, 1993.