Algorithms for Time-Dependent Bicriteria Shortest Path Problems *

Horst W. Hamacher†, Stefan Ruzika†, Stevanus A. Tjandra‡

Abstract

In this paper we generalize the classical shortest path problem in two ways. We consider two objective functions and time-dependent data. The resulting problem, called the time-dependent bicriteria shortest path problem (TdBiSP), has several interesting practical applications, but has not gained much attention in the literature. After reviewing relevant literature we develop a new algorithm for the TdBiSP with non-negative data. Numerical tests show the superiority of our algorithm compared with an existing algorithm in the literature. Furthermore, we discuss algorithms for the TdBiSP with negative data.

Keywords: Multiple criteria optimization; label setting algorithm; time-dependent shortest path problem, bicriteria shortest path problem.

1 Introduction

Shortest path problems are one of the best studied network optimization problems (see e.g. Bertsekas [4], Ahuja et al. [1], and Schrijver [24]) with many applications, such as in transportations (see e.g. Ahuja et al. [2] and the references therein) and evacuation planning (see e.g. Kostreva and Wiecek [15], Yamada [27], and Getachew et al. [11]). In this paper time-dependent bicriteria shortest path problems (TdBiSP) are considered which differ from common shortest path problems in two ways. First, two conflicting and non-commensurate objective functions replace the

*This research has been supported by Deutsche Forschungsgemeinschaft (DFG) grant HA 1737/7 "Algorithmik großer und komplexer Netzwerke" and by New Zealand's Julius von Haast award.

†Fachbereich Mathematik, Universität Kaiserslautern, Kaiserslautern, Germany; E-mail: {hamacher, ruzika}@mathematik.uni-kl.de .

‡Finance and Management Sciences, University of Alberta, Edmonton, Canada; E-Mail: Stevanus.Tjandra@ualberta.ca .

single objective function, and second, the data is subject to changes over time. Throughout this article time is assumed to be discrete unless stated otherwise.

TdBiSP form the basis of many real-world applications (see e.g. Möhring [17] or Müller-Hannemann and Schnee [18]) since many processes are time-dependent and more than one set of time-dependent cost parameters are to be taken into account. Our main motivation is the application in evacuation modelling, where shortest paths represent evacuation routes. Obviously, these routes might change over time. Several attributes associated with a route, like its length or reliability, are of particular interest for an evacuee in case of emergency. Consequently, two or more objectives have to be regarded when planning evacuation routes. A collection of possible evacuation routes could then be used to model a complete evacuation plan. An overview of the interrelation of time-dependent network optimization problems and evacuation modeling can be found in Hamacher and Tjandra [12].

Without time dependency, there are several papers dealing with multicriteria shortest path problems. Hansen [13] introduced ten different types of bicriteria path problems. He also showed that one can construct a specific static network in which the number of Pareto optimal paths grows exponentially with the number of nodes in the network. Martins [16] developed a multiple labeling version of Dijkstra'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 [8] used dynamic programming to solve multicriteria shortest paths problems. Brumbaugh-Smith and Shier [5] proposed a label correcting algorithm with multiple labeling for bicriteria shortest path problems. They also proposed a linear time algorithm for merging two Pareto optimal sets. In their annotated bibliography of multicriteria combinatorial optimization Ehrgott and Gandibleux [10] provide a more complete survey on multicriteria shortest path algorithms.

On the other hand, there are various papers, dealing with time dependency, but with a single objective function. Here, the time-dependent shortest path problem can be classified into fastest path and minimum cost path. In fastest path problems, the cost of an arc is the travel time of that arc. Its objective is then to find paths having minimum length with respect to time-dependent travel times. The minimum cost path problem looks for paths having minimum length with respect to some 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 [7]. Bellman's optimality principle (Bellman [3]) is extended to obtain the fastest paths from every node in the network to one destination node, i.e. *all-to-one* fastest paths. The travel times on the arcs are positive integers for every time period. Dreyfus [9] proposed a modification of Dijkstra's static shortest path algorithm having the same complexity as Dijkstra's algorithm to calculate fastest paths between two nodes for a given departure time. The algorithm works well on FIFO (First-In First-Out) networks, i.e., networks having the non-overtaking property that along every arc (i, j), a unit leaving node i earlier will not arrive at node j later than a unit leaving later. Ziliaskopoulos and Mahmassani [28] and Wardell and Ziliaskopoulos [26] 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 assumptions. Chabini [6] employed a label setting algorithms running backward in the set of time parameters to address both fastest and minimum cost paths for all possible departure times without FIFO requirement. Ahuja et al. [2] investigate the complexity of and propose algorithms for several dynamic minimum cost flow problems. They assume some linear relationship between arc costs and travel time. In a continuous-time environment, several results on time-dependent shortest path problems are included in Orda and Rom [19, 20], Philpott and Mees [23], and Philpott [22].

In the literature, multicriteria dynamic shortest path problems, as combination of the above two problem classes, has gained relatively little attention. Kostreva and Wiecek [15] generalized the result of Cooke and Halsey [7] to the multicriteria case. The cost functions are assumed to be positive vector-valued functions of time and may be discontinuous. A backward dynamic programming approach 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. [11] generalize the results of Kostreva and Wiecek by replacing the non-decreasing arc cost assumption by lower and upper bounds on 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.

The remainder of the paper is organized as follows: In the next section, we formally introduce TdBiSP together with some definitions and notations. In Section 3 we classify the problem as $\mathcal{NP}$-hard, show that Bellman's optimality principle does only hold in a special version. Section 4 shows how to use labels in the time-dependent and bicriteria environment. In Section 5 we assume non-negative data, review an algorithm proposed by Kostreva and Wiecek [15], present a new label setting algorithm, and compare both algorithms numerically. The non-negativity assumption is dropped in Section 6 and we discuss a label correcting approach as well as an approach based on the idea of the Floyd-Warshall algorithm.

2 The Time-Dependent Bicriteria Shortest Path Problem

Consider a discrete-time-dependent network $G = (N, A, T)$ with node set N and arc set A, and finite time horizon $T \in \mathbb{N}$. Let $n = |N|$ and $m = |A|$. Let $d \in N$ be a

designated node, called the sink. Each arc $(i,j) \in A$ is associated with a travel time and cost function each of which might change over time. For each arc $(i,j) \in A$ and each $t \in \{0,\dots,T\}$, let $\lambda_{ij}(t) \in \mathbb{Z}$ denote the travel time of arc (i,j), i.e., the time needed to traverse the arc (i,j) departing from node i at time t. 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* (see Orda and Rom [19]). Note that we do not exclude the possibility that $\lambda_{ij} < 0$. This is, for instance, important in applying our model in solving minimum cost dynamic network flow problems.
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*

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

where $t_{q+1} = t_q' + \lambda_{j_q j_{q+1}}(t_q')$, $q=1,\dots,l-1$ and $t_l := t_l'$.
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. The ready time t_q is equal to the arrival time at node j_q from the previous node j_{q-1}, for $q=2,\dots,l$.

Figure 1 illustrates Definition 2.1.

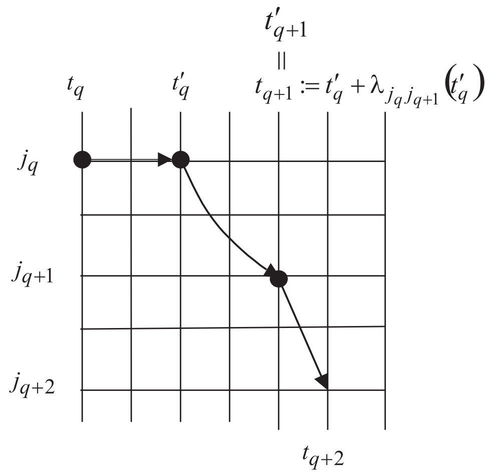


Figure 1: Dynamic Path

Let $\mathbb{P}_{j_1 j_l}(t_1)$ denote 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*

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

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 $t_1^{'} = t_l^{'}$.*

If all travel times are positive, no dynamic path can exist. However, since negative travel times are allowed this may be the case as is illustrated in Example 2.1.

For each arc $(i,j) \in A$ and each $t \in \{1, \dots, T\}$ a two dimensional vector $c_{ij}(t) = \begin{pmatrix} c_{ij}^1(t) \\ c_{ij}^2(t) \end{pmatrix} \in \mathbb{R}^2$ defines the cost of traversing the arc (i,j) starting in node i at time t. In case of waiting $h_i(t) = \begin{pmatrix} h_i^1(t) \\ h_i^2(t) \end{pmatrix} \in \mathbb{R}^2$ specifies the waiting cost for $i \in N$ and $t \in \{0, \dots, T\}$. Here, $h_i^1(t) = \infty$ or $h_i^2(t) = \infty$ indicates that no waiting is allowed. If $t \in \{0, \dots, T\}$ is a ready time and $t^{'}$ a departure time of node i with $t^{'} \geq t$, then we denote

$$H_i(t, t^{'}) := \begin{pmatrix} H_i^1(t, t^{'}) \\ H_i^2(t, t^{'}) \end{pmatrix} = \begin{pmatrix} \sum_{t''=t}^{t^{'}-1} h_i^1(t'') \\ \sum_{t''=t}^{t^{'}-1} h_i^2(t'') \end{pmatrix}.$$

Definition 2.4 *The total cost of $j_1 - j_l$ path $P_{j_1 j_l}(t_1) \in \mathbb{P}_{j_1 j_l}(t_1)$ is*

$$cost(P_{j_1 j_l}(t_1)) := \begin{pmatrix} cost^1(P_{j_1 j_l}(t_1)) \\ cost^2(P_{j_1 j_l}(t_1)) \end{pmatrix} = \sum_{q=1}^{l-1} \begin{pmatrix} c_{j_q j_{q+1}}^1(t_q^{'}) + H_{j_q}^1(t_q, t_q^{'}) \\ c_{j_q j_{q+1}}^2(t_q^{'}) + H_{j_q}^2(t_q, t_q^{'}) \end{pmatrix}.$$

Based on this cost concept, negative dynamic cycles and Pareto optimal paths can be defined.

Definition 2.5 (Negative dynamic cycle) *A dynamic cycle $P_{j_1 j_l}(t_1)$ is a negative dynamic cycle if at least one component of $cost(P_{j_1 j_l}(t_1))$ has a negative value.*

Example 2.1

Consider the network shown in Figure 2, where we assume that no waiting is allowed. Consider the path

$$P_{22}(t) = \{2(t,t), 4(t+3, t+3), 3(t+5, t+5), 2(t+6, t+6)\}$$

for some feasible $t \in \{0, \dots, T\}$. This path passes through node 2 twice, but with different departure and arrival times. Hence, $P_{22}(t)$ is not a dynamic cycle.
Consider the path

$$P_{22}'(t) = \{2(t,t), 4(t+3, t+3), 6(t+2, t+2), 5(t+1, t+1), 2(t,t)\}.$$

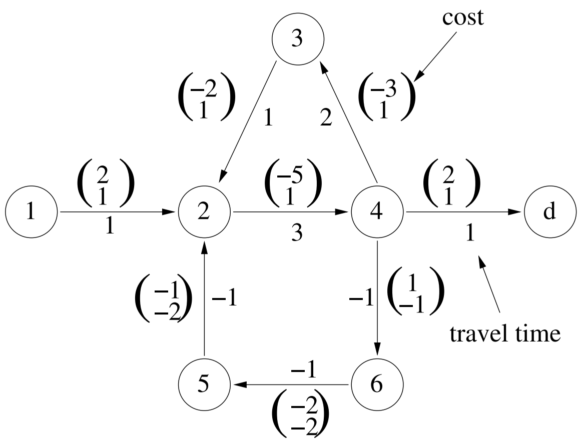


Figure 2: Network with finite Pareto dynamic cycle

This path is a dynamic cycle for all $0 \geq t \leq T-3$. The total cost of path $P'_{22}(t)$ is

$$cost(P'_{22}(t)) = \begin{pmatrix} -7 \\ -4 \end{pmatrix}.$$

Hence, $P'_{22}(t)$ is a negative dynamic cycle. $\square$

The cost of a path is a two-dimensional vector. In general there does not exist a solution minimizing both cost components simultaneously. Thus, the Pareto concept of optimality is used. The following binary relations facilitate the definition of Pareto optimality.

Definition 2.6 *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 2.7 (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*

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

(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 $P_{id}(t)$, $P'_{id}(t) \in \mathbb{P}_{id}(t)$ are equivalent if and only if $cost(P'_{id}(t)) = cost(P_{id}(t))$.*

(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 contain 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 $t \in \{0, \ldots, T\}$ at node i.*

Subsequently we address the problem of finding for all $t \in \{0, \ldots, T\}$ and $i \in N$ a minimal complete set $PO(\mathbb{P}_{id}(t))$ of Pareto paths starting in node i at time t and reaching the sink d within time horizon T.

3 Complexity and Optimality Principles

3.1 Complexity

As in the case of bicriteria static shortest path problems (see Hansen [13]), the number of PO-paths might grow exponentially with the number of nodes in the network in general.

Example 3.1
Consider the network in Figure 3.

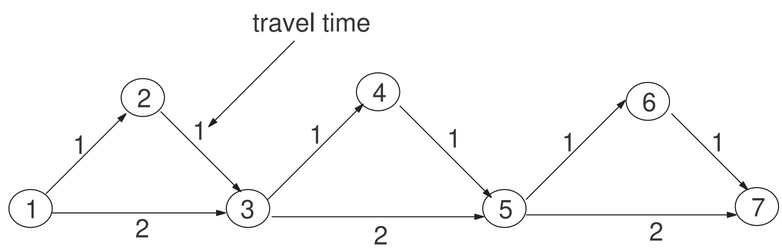


Figure 3: Example network with exponential number of Pareto optimal paths

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

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

We assume that no waiting is allowed for any node. It can be easily verified that there are $2^{\frac{n-1}{2}} = 8$ PO-paths from node 1 to node 7 for every ready time $0 \leq t \leq T - 6$. $\square$

Obviously, this example can be generalized to establish the following result.

Theorem 3.1 *Finding the complete minimal set of Pareto paths is $\mathcal{NP}$-hard.*

3.2 Principle of Optimality

In static shortest path problems, the principle of optimality states that subsets of the optimal solution are also optimal (see e.g. Bellman [3]). We consider the principle of optimality for multicriteria shortest path problems with fixed ready time first in forward and then in backward direction.

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 $PO(\mathbb{P}_{si}(t))$.

Example 3.2 shows that the principle of optimality in forward direction does not hold even under restrictive assumptions on the arcs and cost functions.

Example 3.2
Consider the dynamic network $G = (N, A, T = 8)$ with node 3 being the sink shown in Figure 4. The data of this graph is given by
$\lambda_{01}(t) = 3,\ \lambda_{02}(t) = \lambda_{12}(t) = \lambda_{23}(t) = 1,\ t \geq 0$, and
$c_{01}(t) = c_{12}(t) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, c_{02}(t) = \begin{pmatrix} 5 \\ 6 \end{pmatrix}, c_{23}(t) = \begin{pmatrix} 2t+2 \\ 2t+3 \end{pmatrix},\ t \geq 0.$
No waiting is allowed.

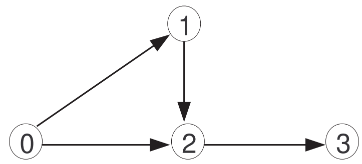


Figure 4: A network for Example 3.2

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

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

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 $t = 0$ and total cost

$$cost(P'_{02}(0)) = \begin{pmatrix} 2 \\ 4 \end{pmatrix}.$$

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

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

$P_{02}(0)$ is a sub-path of a Pareto optimal path $P_{03}(0)$, but it is dominated. Note that in this example all arcs (i,j) fulfill the FIFO property, i.e., the earlier flow leaves node i along arc (i,j), the earlier it arrives. Furthermore, all arcs satisfy the so-called cost consistency property (see e.g. Pallottino and Scutella [21]), i.e., leaving node i earlier along arc (i,j) does not cost more than leaving later. Hence, in general the forward principle of optimality does not hold even if FIFO and cost consistency properties are fulfilled. $\square$

If the forward principle of optimality is not valid, constructing optimal paths based on previously computed paths is not possible in a forward direction. Hence, a forward label setting-type algorithm cannot be used to find Pareto optimal paths. We therefore apply Bellman's principle of optimality in backward direction.

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 $PO(\mathbb{P}_{jd}(t_j))$.

The following theorem shows that the backward principle of optimality holds, even if FIFO or cost consistency assumption is not satisfied.

Theorem 3.2 *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 $P_{id}(t) \in PO(\mathbb{P}_{id}(t))$ and let $j(t_j, t'_j)$ be an arbitrary NTP of $P_{id}(t)$. $P_{id}(t)$ decomposes into sub-paths $P_{ij}(t)$ and $P_{jd}(t_j)$.
Suppose the sub-path $P_{jd}(t_j)$ is not Pareto optimal. Then there is a path $P'_{jd}(t_j)$ with $cost(P'_{jd}(t_j)) < cost(P_{jd}(t_j))$.
Concatenating $P_{ij}(t)$ and $P'_{jd}(t_j)$ yields a path $P'_{id}(t)$ which dominates $P_{id}(t)$ since

$$cost(P'_{id}(t)) = cost(P_{ij}(t)) + cost(P'_{jd}(t_j)) < cost(P_{ij}(t)) + cost(P_{jd}(t_j)) = cost(P_{id}(t)),$$

thus contradicting the fact that $P_{id}(t)$ is nondominated. $\blacksquare$

Theorem 3.2 guarantees that we can use the label setting principles in a backward direction to find *all-to-one* Pareto optimal paths. This will be worked out in more detail in Section 5.

4 Label Structure

In Section 5 and Section 6 we discuss algorithms using labels of the same structure. These labels are introduced in this section. The need to store the time components and two cost components makes the structure of a label more complex than that of single objective static shortest path problems.

We denote by $\pi_i(t)$ a label which corresponds to a path from node i to node d starting at time t. The label $\pi_i(t)$ consists of the following components:

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

The tuple of reals, tuple of integers, and the 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

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

We denote by $cost(\pi_i(t))$ the cost component of label $\pi_i(t)$. $\Pi_i(t)$ denotes in the following a set of labels of node i at time t.

5 Algorithm for All Ready Times with Non-Negative Data

In this section the cost components are restricted to be nonnegative and the travel times must be positive. First we discuss the algorithm of Kostreva and Wiecek [15]. Then we propose a backward label setting algorithm.

5.1 Backward Dynamic Programming

Kostreva and Wiecek [15] developed a backward dynamic programming approach to generate $\mathrm{PO}(\mathbb{P}_{id}(t))$ for all $i \in N$ and all $t \in \{0, \ldots, T\}$. They introduced sets $\Pi_i(t_{ready})^{(l)}$ which contain labels representing a Pareto optimal $i-d$ path of at most l arcs starting at time t. The idea is to compute successively sets of labels $\Pi_i(t)^{(l+1)}$ for every i and t based upon the previously computed sets $\Pi_i(t)^{(l)}$.

Suppose $\Pi_i(t_{ready})^{(l)}$ is given for each node i and ready time t_{ready}. Then, a dynamic path of length $l+1$ starting in node i at time t_{ready} might remain some time in node i until it departs from i at time t_{depart}. After leaving node i, it uses some arc (i,j) and arrives in j at time $t_{arrival} = t_{depart} + \lambda_{ij}(t_{depart})$. The rest of the path is then determined by a path of length l starting in j at time $t_{arrival}$, i.e., by some label in $\Pi_i(t_{arrival})^{(l)}$.

Therefore, when determining $\Pi_i(t_{ready})^{(l+1)}$, all labels $\Pi_j(t_{arrival})^{(l)}$ with $(i,j) \in A$ and $t_{arrival} = t_{depart} + \lambda_{ij}(t_{depart})$ have to be considered. In doing so, t_{depart} is taking on all values $t_{ready}, t_{ready}+1, \ldots$ as long as waiting is permitted. Hence,

$$\begin{aligned} Depart_i(t_{ready}) \quad := \quad & \Big\{ t : t_{ready} \leq t \leq t^* \text{ with} \\ & \quad t^* = \min\left\{ t' : t' \geq t_{ready},\ h_i^k(t') = \infty,\ k \in \{1,2\}\right\}\Big\} \end{aligned}$$

is the set of all possible departure times.

Since we are only interested in Pareto optimal solutions in each step, we denote by $VMIN(\Pi_j(t))$ the vector minimization with respect to the cost component over all elements of the label set $\Pi_j(t)$. This process yields the set of labels with associated Pareto optimal costs. Formally, the set $\Pi_i(t_{ready})^{(l+1)}$ is obtained for any $t_{ready} \in \{0,\ldots,T\}$ by

$$\Pi_i(t_{ready})^{(l+1)} = \begin{cases} VMIN\big(\Pi_j(t_{arrival})^{(l)} + c_{ij}(t_{depart}) + \\ \qquad\qquad H_i(t_{ready}, t_{depart})\big), \ i \in N \setminus \{d\} \\ \left\{ \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} ; (t_{ready}, t_{ready}); \emptyset \right) \right\}, \ i = d \end{cases} \tag{3}$$

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

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

adds the waiting cost $H_i(t_{ready}, t_{depart})$ and the (travel) cost $c_{ij}(t_{depart})$ to the cost component of each label in $\Pi_j(t_{arrival})^{(l)}$. This operation also sets the 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. This iterative process continues until $\Pi_i(t_{ready})^{(l+1)} = \Pi_i(t_{ready})^{(l)}$ for some $l \in \mathbb{N}$ and all $i \in N$ and $t_{ready} \in \{0,\ldots,T\}$. The complete algorithm is summarized in Algorithm 5.1.

5.2 Backward Label Setting Algorithm

Since all data are assumed to be non-negative we extend the idea of Dijkstra's label-setting static shortest path algorithm. Theorem 3.2 guarantees that the labeling

Algorithm 5.1 Kostreva and Wiecek [15]

INPUT	Network $G = (N, A, T)$; $c(t)$ and $h(t)$ $\forall t \in \{0, \ldots, T\}$.
OUTPUT	$PO(\mathbb{P}_{id}(t))$, $\forall i \in N$, $\forall t \in \{0, \ldots, T\}$.
0	$c_{ij}(t) = \begin{cases} \begin{pmatrix} c^1_{ij}(t) \\ c^2_{ij}(t) \end{pmatrix} & , t + \lambda_{ij}(t) \le T \\ \begin{pmatrix} \infty \\ \infty \end{pmatrix} & , t + \lambda_{ij}(t) > T \end{cases} \quad ; \ \forall (i,j) \in A, \forall t \in \{0, \ldots, T\}$
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 in (3).
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.

process can be done in a backward direction from the sink node d to all other nodes in the network.

We introduce label sets for each node $i \in N$ at each ready time $t_{ready} \in \{0, \ldots, T\}$: the set of *temporary* and the set of *permanent labels* denoted by $\Pi_i(t_{ready})_{prm}$ and $\Pi_i(t_{ready})_{tmp}$, respectively. The set $\Pi_i(t_{ready})_{prm}$ contains labels representing non-equivalent Pareto optimal paths which can be constructed by backtracking the successor pointers *succ_ptr*. Each of these Pareto paths starts in node i at time t_{ready} and uses only nodes which have been permanently labeled as well. Once computed, the elements of this set remain unchanged. The set $\Pi_i(t_{ready})_{tmp}$ contains labels representing candidates for Pareto optimal paths. Labels therein might be changed or deleted.

Initially, we define $\Pi_i(t_{ready})_{prm}$ to be the empty set for each $i \in N$ and each $t_{ready} \in \{0, \ldots, T\}$. Since we assume that no arc leaves the sink node d, we assign to the sink node d at any time $t_{ready} \in \{0, \ldots, T\}$ a temporary label having cost component zero. The empty set is assigned to every other node i.

$$\Pi_i(t_{ready})_{tmp} \ := \ \begin{cases} \left\{ \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} ; (t_{ready}, t_{ready}); \emptyset \right) \right\} & , i = d \\ \emptyset & , \text{otherwise} \end{cases} , \quad t_{ready} \in \{0, \ldots, T\} \tag{4}$$

In each iteration a suitable label - called the pivot label - is moved from some set

$\Pi_j(t_{arrival})_{tmp}$ to the set $\Pi_j(t_{arrival})_{prm}$. Using this pivot label, candidate labels for sets $\Pi_i(t_{ready})_{tmp}$ are generated by the following rules:

- there is an arc connecting nodes i and j, i.e. $(i, j) \in A$,
- waiting has to be allowed in node i beginning with time t_{ready} until some t_{depart}, and
- the travel time has to fulfill $t_{arrival} = t_{depart} + \lambda_{ij}(t_{depart})$.

How to choose this pivot label (this includes the choice of j and $t_{arrival}$) will be shown later. Let us first concentrate on the generation of candidate labels.

Suppose the label $\pi_j^*(t_{arrival}) \in \Pi_j(t_{arrival})_{tmp}$ is chosen to be the pivot label. $\pi_j^*(t_{arrival})$ is deleted from $\Pi_j(t_{arrival})_{tmp}$ and inserted in $\Pi_j(t_{arrival})_{prm}$. Suppose we have found some arc $(i, j) \in A$, and times t_{ready} and t_{depart} as mentioned in the rules above. Then, one new candidate label $\hat{\pi}_i(t_{ready})$ can be generated as follows:

$$\begin{aligned} \hat{\pi}_i(t_{ready}) \;&:=\; \pi_j^*(t_{depart} + \lambda_{ij}(t_{depart})) + (c_{ij}(t_{depart}) + H_i(t_{ready}, t_{depart})), \\ &\quad\;\; \text{where} \qquad (i, j) \in A,\; t_{depart} + \lambda_{ij}(t_{depart}) = t_{arrival}. \end{aligned} \tag{5}$$

In the computation of $\hat{\pi}_i(t_{ready})$, the summation of the cost components, the settings of the time components and the successor pointers are performed analogously to the corresponding operation in the algorithm of Kostreva and Wiecek [15].

In order to address the problem of generating *all* possible candidates for some pivot label, we introduce for all arcs $(i, j) \in A$ the following sets to identify all times t_{depart} and t_{ready} according to the rules above.

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

$$Arrival_{ij}^{-1}(t_{arrival}) \;:=\; \{t' \;:\; t' + \lambda_{ij}(t') = t_{arrival}\}. \tag{6}$$

- $Depart_i^{-1}(t_{depart})$: Given $t_{depart} \in \{0, \ldots, T-1\}$, $Depart_i^{-1}(t_{depart})$ denotes the set of all possible ready times at node $i \in N \setminus \{d\}$ such that waiting in node i starting with t_{ready} until t_{depart} is allowed, i.e.,

$$\begin{aligned} Depart_i^{-1}(t_{depart}) \;:=\; \Big\{ & t : t^* < t \leq t_{depart} \text{ with} \\ & t^* = \max\left\{t' : t' \leq t_{depart},\; h_i^k(t') = \infty,\; k \in \{1, 2\}\right\}\Big\}. \end{aligned}$$

While the set $Arrival_{ij}^{-1}(t_{arrival})$ may be empty, the set $Depart_i^{-1}(t_{depart})$ has cardinality of at least one, since $t_{depart} \in Depart_i^{-1}(t_{depart})$. Given the pivot label $\pi_j^*(t_{arrival})$, all possible candidates can then be computed by considering all ready times in $Depart_i^{-1}(t_{depart})$ for each departure time in $Arrival_{ij}^{-1}(t_{arrival})$.

The candidate label $\hat{\pi}_i(t_{ready})$ is then merged with $\Pi_i(t_{ready})_{tmp}$. This means, if $\hat{\pi}_i(t_{ready})$ is dominated by some label in $\Pi_i(t_{ready})_{tmp}$, the candidate $\hat{\pi}_i(t_{ready})$ is deleted. Otherwise all labels in $\Pi_i(t_{ready})_{tmp}$ dominated by $\hat{\pi}_i(t_{ready})$ are deleted and $\hat{\pi}_i(t_{ready})$ is inserted in $\Pi_i(t_{ready})_{tmp}$. We denote this merging process by $Merge(\{\hat{\pi}_i(t_{ready})\}, \Pi_i(t_{ready})_{tmp})$.

The algorithm iteratively chooses a pivot label, sets its label as permanent, computes candidate labels, and merges the candidate labels with perviously computed labels until $\Pi_i(t_{ready})_{tmp} = \emptyset$ for all $i \in N$ and all $t_{ready} \in \{0, \ldots, T\}$. Then, the algorithm terminates.

The problem of choosing a suitable pivot element is addressed by the following theorem in which the permanence of the labels is established if the labeling process is done starting with $t = T$ and ending with $t = 0$, i.e, with decreasing parameter t.

Theorem 5.1 *For each node i and time t_{ready}, the labeling process of label $\pi_i(t_{ready})$ depends on labels at time $t > t_{ready}$ but does not depend on labels of times $t \leq t_{ready}$.*

Proof:
Since $\lambda_{ij}(t_{depart}) > 0$, (5) includes only $\pi_j(t_{arrival})$ for $t_{arrival} > t_{depart}$. ■

This theorem implies that pivot elements are chosen from the set $\Pi_j(t_{arrival})_{tmp}$ for any $j \in N \setminus \{d\}$ only when $\Pi_j(t_{arrival} + 1)_{tmp} = \emptyset$ for all $j \in N$ and $t_{arrival} \in \{0, \ldots, T-1\}$, i.e, only when all labels of $\Pi_j(t_{arrival})_{tmp}$ have already been processed. Let us assume in the following that $t_{arrival}$ is the processing time currently of interest. We have therefore proved the following result:

Corollary 5.1 *If $\Pi_j(t_{arrival} + t)_{tmp} = \emptyset$ for all $j \in N$ and $t = 1, \ldots, T - t_{arrival}$, then for any $i \in N$ any $\pi_j^*(t_{arrival}) \in \Pi_j(t_{arrival})_{tmp}$ can be chosen as pivot label.*

The detailed description on how to find the minimal complete set of all PO-paths from every node $i \in N$ to the sink d for all (fixed) ready times $t \in \{0, \ldots, T\}$ is given in Algorithm 5.2. At the end of algorithm permanent labels associated with node $i \in N$ and ready time $t \in \{0, \ldots, T\}$ correspond to non-equivalent Pareto optimal dynamic paths from i with ready time t to the sink node. The correctness of Algorithm 5.2 is established by the following theorem.

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

Proof:
Initially, all labels corresponding to Pareto optimal paths starting in node d at time t for all $t \in \{1, \ldots, T\}$ are found and marked as permanent. Theorem 5.1 and Proposition 5.1 ensure that permanent labels do not change anymore with decreasing time parameter t. Based on the set of permanent labels, new candidate labels are generated. Theorem 3.2 ensures that in Step 2 of the algorithm, all possible candidates for Pareto optimal labels are tested. The $Merge$-operator filters dominated labels. Finally, a set $\Pi_i(t)_{tmp}$ is Pareto optimal, complete, and - since the merging is done with respect to the cost components - also minimal. $\blacksquare$

Algorithm 5.2

INPUT	Network $G = (N, A, T)$; $c(t)$ and $h(t)$ $\forall t \in \{0, \ldots, T\}$.
OUTPUT	$PO(\mathbb{P}_{id}(t))$, $\forall i \in N$, $\forall t \in \{0, \ldots, T\}$.
0	$c_{ij}(t) = \begin{cases} \begin{pmatrix} c^1_{ij}(t) \\ c^2_{ij}(t) \end{pmatrix} & , t + \lambda_{ij}(t) \leq T \\ \begin{pmatrix} \infty \\ \infty \end{pmatrix} & , t + \lambda_{ij}(t) > T \end{cases} \quad ; \forall (i,j) \in A, \forall t \in \{0, \ldots, T\}$
1	Define the initial set of temporary labels to every node $i \in N$ and for all $t \in \{0, \ldots, T\}$ 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 $t \in \{0, \ldots, T\}$, i.e. $\Pi_i(t)_{prm} := \emptyset$.

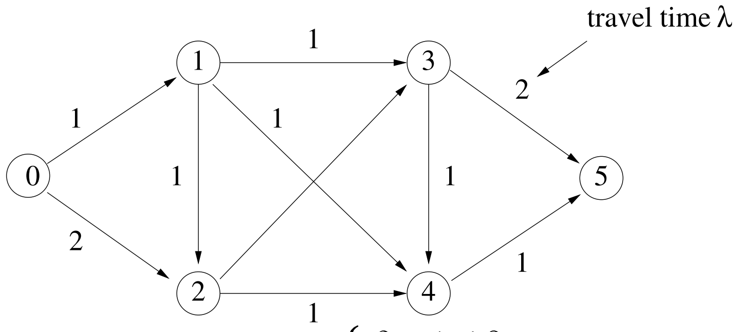
2	For $t_{arrival} = T$ down to 0 do
	{
	While $\bigcup_{i \in N} \Pi_i(t_{arrival})_{tmp} \neq \emptyset$ do
	{ Select a pivot label $\pi^*_j(t_{arrival})$ from $\Pi_j(t_{arrival})_{tmp}$
	Remove $\pi^*_j(t_{arrival})$ from $\Pi_j(t_{arrival})_{tmp}$ to $\Pi_j(t_{arrival})_{prm}$.
	For all $(i,j) \in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

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

The following example illustrates Algorithm 5.2.

Example 5.1

Consider the network in Figure 5 with sink node 5 and $T = 8$. Cost data corre-



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

Figure 5: Network for Example 5.1

sponding to the travel cost c and waiting cost h for each arc and for any time t in $\{0, \ldots, T\}$ are given below.

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

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

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

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

$$H_i(t, t') = \begin{cases} \begin{pmatrix} t' - t \\ t' - t \end{pmatrix} & , \ \forall t' \geq t \\ \begin{pmatrix} \infty \\ \infty \end{pmatrix} & , \ \text{otherwise} \end{cases}$$

The final label set $\Pi_i(t)_{prm}$ for every node $i \in N$ and for every time $t \in \{0, \ldots, T\}$ are reported in Table 1.

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

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

node 0 at time $t = 0$ are

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

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

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

Hence,

$$\begin{aligned} PO(\mathbb{P}_0(0)) &= \big\{\{0(0,0), 2(2,2), 3(4,4), 5(6,6)\}, \{0(0,0), 2(2,2), \\ &\quad 3(3,3), 4(4,4), 5(5,5)\}\big\} \end{aligned}$$

We see from Table 1 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. $\square$

5.3 Computational Results

To compare the performance of Algorithm 5.1 and Algorithm 5.2 numerical experiments were conducted. Both algorithms were run on the same series of randomly generated dynamic networks. A random network generator was developed to produce connected dynamic networks. We extended the idea of NETGEN [14] - a random generator for static networks - to include time-dependent data.

Random networks were generated with n= 50, 100, 500, and 1000 nodes and time horizon $T = 100$. For each choice of n, nodes had indegree and outdegree of 2, 4, 6, and 8, respectively. In each network the sink node had no outgoing arcs, i.e., the sink node's outdegree was zero. All data were integer. The travel time was randomly chosen from the set $\{1, \dots, 10\}$. Both cost components and waiting costs were at least 0 and at most 10. For each specific setting of n and m, we tested five random dynamic networks. Therefore, the total number of observations is 80.

Algorithms 5.1 and 5.2 were implemented in C++. The testing was performed on a PC equipped with a 500 MHz Pentium III processor and 256 MB RAM. The results of the experiments are given in Table 2. According to the average running

n	m/n	Average CPU-time (in seconds	
		Algorithm 5.1	Algorithm 5.2
50	2	2.66	1.01
	4	6.00	1.64
	6	6.24	2.29
	8	14.47	3.00
100	2	7.37	2.24
	4	14.61	3.65
	6	24.63	5.18
	8	36.98	6.90
500	2	62.76	14.61
	4	164.99	24.42
	6	321.29	35.83
	8	464.51	48.69
1000	2	251.85	31.41
	4	756.21	55.99
	6	1,269.37	83.87
	8	2,275.02	117.09

Table 2: Computational test results

times, Algorithm 5.2 outperforms Algorithm 5.1 in all settings. This shows the superiority of the label setting approach over the dynamic programming approach in the problem of finding the minimal complete set $PO(\mathbb{P}_{id}(t))$.

6 Extensions

In the following we address the case when negative data is allowed. Since the existence of negative dynamic cycles can no longer be excluded, we study the effects of negative dynamic cycles and discuss solution concepts. Furthermore, we study dynamic path problems defined by their arrival time instead of their starting time. For this problem, the forward principle of optimality holds (see Definition 3.1).

6.1 Negative Data

If negative travel times and costs are allowed, negative dynamic cycles may exist (see Example 2.1). As in the static case, Dijkstra's label setting algorithm is not capable of handling negative data since dominating paths might not be recognized and negative cycles cannot be detected. Therefore, an algorithm is required which recognizes negative dynamic cycles.

Recall the definition of negative dynamic cycle: a dynamic cycle is negative if at least one of the cycle's cost components is negative. As stated in Theorem 6.1, the existence of a negative cycle might have two different consequences for the solution sets of the problem.

Theorem 6.1 *Let $i \in N$ and $t \in \{0, \ldots, T\}$. Suppose there is an $i-d$ path P with ready time t, containing a negative dynamic cycle.*

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

(b) *If P is Pareto optimal and if one cost component of the negative dynamic cycle is positive, then there will be an infinite number of Pareto optimal dynamic $i-d$ paths.*

Proof:
Let C be a negative dynamic cycle in the $i-d$ path P.

(a) Repeating the cycle once yields another path P' that dominates P since the cycle has one cost component being strictly negative and the other being non-positive. Using this argument iteratively shows that $PO(\mathbb{P}_{id}(t)) = \emptyset$.

(b) Repeating the cycle once yields a new path P' which neither dominates nor is dominated by P since one cost component is improved and the other is worsened. Since the cycle can be repeated arbitrarily often, the number of Pareto optimal dynamic $i-d$ paths is not bounded. ■

In Case (a) of Theorem 6.1 the problem is ill-posed. In contrast, a negative dynamic cycle as in Case (b) implies that the optimal set is unbounded. Nevertheless, the set of Pareto optimal paths defined by repetitions of negative dynamic cycles can be expressed using periodicity. In view of Theorem 6.1, a suitable algorithm should not only detect the presence of a negative dynamic cycle, but it should also be able to classify it with respect to Case (a) and Case (b).

It is possible to formulate a label correcting algorithm using a similar label update as in the label setting algorithm of Section 5.2: In each pass, pivot labels $\pi_j^*(t_{arrival})$ are considered according to the first node j in a list of eligible nodes, that is a list of nodes which can potentially yield further label improvements. Each pivot label creates sets of candidate labels $\hat{\Pi}_i(t_{ready})$ for several nodes i and times t_{ready} which are then merged with the existing label set $\Pi_i(t_{ready})$. If this merging process yields newly found labels, node i is inserted into the list of eligible nodes. For a detailed description of a label correcting algorithm the reader is referred to Tjandra [25]. This label correcting algorithm detects the presence of negative cycles either if the cost component of some label is less than $-n(T+1)\cdot\begin{pmatrix} C^1 \\ C^2 \end{pmatrix}$, where

$$C^l = \max_{(i,j)\in A,\ t\in\{0,\ldots,T\}} |c_{ij}^l(t)|, \quad l = 1, 2,$$

or if the number of passes exceeds $(n-1)(T+1)$. In both cases classifying the negative dynamic cycle needs substantial computational effort.

Another way of recognizing and classifying negative dynamic cycles is the generalization of the Floyd-Warshall algorithm see e.g. [1]) to bicriteria dynamic shortest path problems: Instead of label sets associated with nodes, we now consider label sets $\Pi_{ij}^k(t_{ready})$ associated with each node pair $(i,j) \in N \times N$ and $t_{ready} \in \{0,\ldots,T\}$. The elements of a set $\Pi_{ij}^k(t_{ready})$ correspond to Pareto optimal paths starting from node i at time t_{ready} and ending at node j where only nodes $1,\ldots,k-1$ are used along this path. In iteration $k+1$, candidate labels are found by combining the labels of set $\Pi_{ik}^k(t_{ready})$ and $\Pi_{kj}^k(t_{ready}^k)$ where t_{ready}^k is the ready time of a $k-j$ path. Merging the candidate labels with $\Pi_{ij}^k(t_{ready})$ yields $\Pi_{ij}^{k+1}(t_{ready})$. Note, that the ready time t_{ready}^k of a $k-j$ path depends on the arrival time of the $i-k$ path in node k. However, t_{ready}^k is not known. Moreover, the idea of Floyd-Warshall is based on the fact that Pareto optimal $i-j$ paths using node k as an intermediate node are always composed of Pareto optimal $i-k$ paths and Pareto optimal $k-j$ paths. But for $PO(\mathbb{P}_{id}(t_{ready}))$ the forward principle of optimality is not valid. These problems are further discussed in the following subsection.

6.2 General Shortest Path Problems

So far we have considered the problem of finding $PO(\mathbb{P}_{id}(t_{ready}))$ for every $i \in N$ and $t_{ready} \in \{0,\ldots,T\}$, that is the minimal complete set of all Pareto optimal paths starting from node i at time t_{ready} and arriving at the sink d before time T. In this problem the path's ready time in node i is fixed, whereas the path's arrival time in sink node d is not specified.

Depending on whether a path's starting and arrival time is fixed or arbitrary, the following four problems can be formulated: For each $i \in N$, find

- $PO(\mathbb{P}_{id})$,
  i.e., a minimal complete Pareto set of $i-d$ paths with arbitrary ready time in node i and arbitrary arrival time in node d, or

- $PO(\mathbb{P}_{id}(t_{ready}))$,
  i.e., a minimal complete Pareto set of $i-d$ paths with fixed ready time t_{start} in node i and arbitrary arrival time in node d, or

- $PO(\mathbb{P}_{id}(t_{arrival}))$,
  i.e., a minimal complete Pareto set of $i-d$ paths with arbitrary ready time in node i and fixed arrival time t_{end} in node d, or

- $PO(\mathbb{P}_{id}(t_{ready}, t_{arrival}))$,
  i.e., a minimal complete Pareto set of $i-d$ paths with fixed ready time t_{start} in node i and fixed arrival time t_{end} in node d.

Depending on which of the preceding cases applies, the notion of Pareto optimality changes. Analogous to the considerations in Section 3.2 it can then be shown that for these problems the results concerning the validity of the forward and backward principle of optimality summarized in the Table 4 can be obtained. Since the proofs

Problem	Forward Principle	Backward Principle
$PO(\mathbb{P}_{id})$	no	no
$PO(\mathbb{P}_{id}(t_{ready}))$	no	yes
$PO(\mathbb{P}_{id}(t_{arrival}))$	yes	no
$PO(\mathbb{P}_{id}(t_{ready}, t_{arrival}))$	yes	yes

Table 3: Optimality Principles and General Bicriteria Dynamic Shortest Path Problems

and counterexamples follow the ideas from Section 3.2, they are left to the reader.

6.3 A Modified Floyd-Warshall Algorithm

We are now in the position to overcome the difficulties from Section 6.1 by solving an all-to-all bicriteria dynamic shortest path problem with fixed ready and fixed arrival times. Note that

$$\mathbb{P}_{ij}(t_{ready})) = \bigcup_{t_{arrival} \in \{0,\dots,T\}} \mathbb{P}_{ij}(t_{ready}, t_{arrival}))$$

Let us consider the generation of candidate labels in iteration $k+1$. For all possible arrival times $t^k_{arrival}$ each label in $\Pi^k_{ik}(t_{ready}, t^k_{arrival})$ is combined with each label in $\Pi^k_{kj}(t^k_{ready}, t^j_{arrival})$ for every possible arrival time $t^j_{arrival}$ in node j to produce candidates. The choice of the ready time of the $k-j$ path depends on the arrival time of the $i-k$ path and the possibility of waiting in node k. The waiting cost between $t^k_{arrival}$ and t^k_{ready} has to be added to the candidate label's cost component. The validity of the forward and the backward principle of optimality justifies this approach.

If there is some label $\pi^k_{ii}(t_{ready}, t_{arrival}) \in \Pi^k_{ii}(t_{ready}, t_{arrival})$ for some $i \in N$ with $t_{ready} = t_{arrival}$ then we have found a negative dynamic cycle. Furthermore, this negative cycle can be easily classified according to Theorem 6.1.

References

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

[2] R. Ahuja, J. Orlin, S. Pallottino, and M. Scutella. Dynamic shortest paths minimizing travel times and costs. *Networks*, 41:197–205, 2003.

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

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

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

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

[7] 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.

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

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

[10] M. Ehrgott and X. Gandibleux, editors. *Multiple Criteria Optimization. State of the Art Annotated Bibliographic Surveys.* Kluwer, Boston, 2002.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

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

[17] R. H. Möhring. Verteilte Verbindungssuche im öffentlichen Personennahverkehr: Graphentheoretische Modelle und Algortihmen. Technical Report 624-1999, Technische Universitt Berlin, 1999.

[18] M. Müller-Hannemann and M. Schnee. Finding all attractive train connections by multi-criteria pareto search. In *Proceedings of the 4th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2004), Bergen, Norway (to appear)*, Lecture Notes in Computer Science, Springer.

[19] 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.

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

[21] 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.

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

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

[24] A. Schrijver. *Combinatorial Optimization. Polyhedra and Efficiency.* Springer, Berlin Heidelberg, 2003.

[25] S. Tjandra. *Dynamic Network Optimization with Application to the Evacuation Problem.* PhD thesis, Technische Universität Kaiserslautern, Kaiserslautern, Germany, 2003.

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

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

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