J Optim Theory Appl (2017) 174:858–874
DOI 10.1007/s10957-017-1145-9

Shortest Paths with Shortest Detours

A Biobjective Routing Problem

Carolin Torchiani[1] · **Jan Ohst**[1] · **David Willems**[1] · **Stefan Ruzika**[1]

Received: 2 March 2016 / Accepted: 17 July 2017 / Published online: 25 July 2017

Abstract This paper is concerned with a biobjective routing problem, called the shortest path with shortest detour problem, in which the length of a route is minimized as one criterion and, as second, the maximal length of a detour route if the chosen route is blocked is minimized. Furthermore, the relation to robust optimization is pointed out, and we present a new polynomial time algorithm, which computes a minimal complete set of efficient paths for the shortest path with shortest detour problem. Moreover, we show that the number of nondominated points is bounded by the number of arcs in the graph.

Keywords Networks · Shortest paths · Biobjective programming · Detours · Recovery robustness

Mathematics Subject Classification 90C27 · 90C29 · 90C35

✉ Carolin Torchiani
torchiani@uni-koblenz.de

Jan Ohst
ohst@uni-koblenz.de

David Willems
davidwillems@uni-koblenz.de

Stefan Ruzika
ruzika@uni-koblenz.de

[1] Mathematisches Institut, Universität Koblenz-Landau, Universitätsstraße 1, 56070 Koblenz, Germany

1 Introduction

The article is motivated by an application situated in the poles of shortest paths, robust optimization and biobjective optimization. Emergency service vehicles, such as ambulances, want to reach their destination as fast as possible. At first glance, it seems to be reasonable to use a quickest route to reach the destination. However, it might happen that the chosen route is blocked by some unforeseen event, e.g., an accident or traffic jam, and the emergency service vehicle has to adapt its route. Taking into account the length (or, alternatively, temporal duration) of the worst-case detour motivates a second criterion for choosing an optimal route in this decision context.

The concept of blocking also appears in the framework of production planning: A production chain can be modeled as a graph, where each arc represents a production substep and the arc costs stand for the money needed to perform the corresponding substep. Again, it seems reasonable to choose the cheapest production path, unless unforeseen blockings are taken into consideration.

The one-to-one shortest path problem in a (directed) graph is a well-studied optimization problem both in the single- and in the multiobjective case. The classical single-objective problem [1,2] can be solved in polynomial time, while the standard multiobjective version of the shortest path problem [3–5] with a cost vector on each arc and sum objective functions is NP-complete [6] and intractable [7].

In robust variants [8] of the shortest path problem [9–12], in which the cost of an arc lies in an interval or in a discrete set, a shortest path with respect to a worst-case scenario or a so-called maximum regret criterion is calculated. All these variants are NP-hard, except for the worst-case interval scenario, which reduces to the single-objective shortest path problem.

2 Preliminaries and Problem Formulation

2.1 Preliminaries

Not only shortest paths, but also detours have been studied extensively. Several authors [13–15] deal with the calculation of the detour-critical arc of a graph $G = (V, A)$ with distinguished node $t \in V$: The detour of an arc $(i, j) \in A$ with $i, j \in V$ is defined as the difference between the cost of a shortest path from i to t and the cost of a shortest path from i to t that does not use the arc (i, j). The detour-critical arc of a graph is the arc with the largest detour. Other authors compute the most vital arc or the most vital node of a graph with distinguished nodes $s, t \in V$ [16–18]: The most vital arc (node) is the arc (node) whose removal maximizes the length of a path from s to t. A generalization is the network interdiction problem [19,20]: Which set of arcs, satisfying a limited interdiction budget, should be removed from the graph in order to maximize the length of a shortest path from s to t?

Moreover, detours have been considered in the context of recoverable robustness [21]. The recoverable robust shortest path problem that is most closely linked to the shortest path with shortest detour problem is the k-Canadian Traveler Problem [22,23], where $k \in \mathbb{N}$ is a natural number. Here, a strategy is sought that guarantees

the shortest worst-case travel time between two fixed vertices if at most k roads turn out to be blocked. This problem is shown to be PSPACE-complete if k is nonconstant [23].

The problem studied in this article combines the classical single-objective shortest path problem with the recoverable robust Canadian Traveler Problem, leading to a biobjective shortest path problem minimizing both objectives. In contrast to the Canadian Traveler Problem, not only the length of a worst-case detour of a path, but also the length of the path itself is minimized.

The remainder of the article is structured as follows. In Sect. 2.2, we formally introduce the biobjective *shortest path with shortest detour problem* (SPSDP). Moreover, we motivate the choice of the second objective function measuring the detour cost of a path. An algorithm for SPSDP is presented in Sect. 3. We show its correctness and a polynomial running time. Furthermore, we prove that the number of nondominated points is bounded by the number of arcs. We conclude the article by discussing further research directions in Sect. 4.

2.2 Problem Formulation

Let $G = (V, A)$ be a finite graph, directed or undirected, with set of vertices V and set of arcs A. Since this problem originates from the application in street networks, in the following, we assume that G does not have parallel arcs. In order to simplify notation, we speak in both the directed and the undirected case of arcs and use the notation $(i, j) \in A$. Let $s, t \in V$ be two distinct vertices of G, which we want to link by a shortest path with shortest detour. For $i, j \in V$, we denote by $\mathcal{P}_{i,j}(G)$ the set of simple paths in G from $i \in V$ to $j \in V$, and we denote by $\mathcal{P}(G)$ the set of all simple paths in G. A path $P \in \mathcal{P}_{i,j}(G)$ is called i-j-path. The travel costs on the arcs of G are given by $c : A \to \mathbb{R}_{\geq 0}$. We define the cost of a path $P \in \mathcal{P}(G)$ as $c(P) := \sum_{(i,j)\in P} c(i, j)$. In case that $P \in \mathcal{P}_{s,t}(G)$, we say that P is a shortest s-t-path if P is an s-t-path of minimal cost. If $i, j \in V$ are two vertices that lie on P and if i lies in front of j, we define $P_{i,j}$ as the path given by the subpath of P that starts at i and ends at j.

We introduce an additional cost function $d_c : A \to [0, \infty]$, which may depend on c, and we call $d_c(i, j)$ the detour cost of $(i, j) \in A$. The function d_c measures the detour cost of an arc: The worst that can happen, when traveling on the arc (i, j), is that the arc turns out to be blocked directly in front of its end vertex j. In this case, one has already traveled all the way from i to j and has to traverse the whole arc back to i in order to find an alternative route toward the destination t. The term $d_c(i, j)$ indicates the cost of traveling back from j to i (even though the arc $(j, i) \notin A$ might be missing in G) and proceeding to t without traversing (i, j). Note that we allow $d_c(i, j) = \infty$, which models among others that it is impossible to traverse the arc (i, j) backward.

Now, let $c' : A \to [0, \infty]$ be a cost function. For $(i, j) \in A$, let $c'(i, j)$ model the cost of turning at j and traveling from j to i. If (i, j) is a one-way road, the arc cost $c'(i, j)$ might stand for the time that an ambulance needs to turn at j and to go backward from j to i. We allow $c'(i, j) = \infty$, which models that it is impossible to go back from j to i. Then, one of the most prominent examples of a detour cost function $d_c : A \to [0, \infty]$ is given by

$$d_c^1(i,j) := c'(i,j) + \min_{P \in \mathcal{P}_{i,t}(G-(i,j))} c(P) \tag{1}$$

for all $(i,j) \in A$, where we set $\min_{P \in \mathcal{P}_{i,t}(G-(i,j))} c(P) := \infty$ if the set $\mathcal{P}_{i,t}$ is empty.

The term $d_c^1(i,j)$ stands for the cost of turning at j, traveling back to i and then proceeding on the shortest path to t that does not use arc $(i,j) \in A$.

Example 2.1 We give a more general example for the detour function d_c, which is motivated by the Canadian Traveler Problem [22,23] and recovery robust optimization:

We consider the scenario that at most $k \in \mathbb{N}$ arcs of the graph may be blocked, but it is a priori unknown which ones. The set of possible realization scenarios is hence parametrized by the uncertainty set

$$\mathcal{U} = \{\delta \in 2^A : |\delta| = k\},$$

where 2^A denotes the power set of A. By $c' : A \to [0,\infty]$ we denote, as above, the turning cost of an arc, i.e., $c'(i,j)$ denotes the cost of going backward from j to i. As above, the tuple (j,i) is not necessarily an element of A.

When traveling on a path and encountering at most k blockings, we use the following recovery algorithm $\mathcal{A}^k$ in order to reach the destination regardless of the blockings:

- When encountering a blocking at the end of arc $(i,j) \in A$, go back to the last junction spending the turning cost $c'(i,j)$.
- Proceed, not using the arc (i,j), to t on a shortest path with shortest detour expecting one blocking less.
- When having encountered k blockings, continue on the shortest path to the destination t.

Using the recovery algorithm $\mathcal{A}^k$, the worst-case detour cost $d_c^k(i,j) \in [0,\infty]$ of an arc $(i,j) \in A$ is given by the cost of the path that minimizes the worst-case travel cost when

- encountering a blocking at j,
- using the recovery algorithm $\mathcal{A}^{k-1}$ to reach the destination k and
- encountering on the way $k-1$ more blockings.

The problem of calculating the detour costs $d_c^k(i,j)$ is the k-Canadian Traveler Problem, which is PSPACE-complete [23].

However, if $k = 1$, i.e., if only one blocking is expected, it holds for $(i,j) \in A$ that

$$d_c^1(i,j) = c'(i,j) + \min_{P \in \mathcal{P}_{i,t}(G-(i,j))} c(P),$$

and all values $d_c^1(i,j)$ can be calculated in time $\mathcal{O}\left(|A| \cdot \left(|V| \log |V| + |A|\right)\right)$ using Dijkstra's algorithm [24].

The two objective functions that we consider are

$$\begin{aligned} f_1 : \mathcal{P}_{s,t}(G) &\to \mathbb{R} \\ P &\mapsto c(P) \end{aligned}$$

and

$$f_2 : \mathcal{P}_{s,t}(G) \to \mathbb{R} \cup \{\infty\}$$
$$P \mapsto \max_{(i,j)\in P} \big(c(P_{s,j}) + d_c(i,j)\big).$$

The function f_1 evaluates the travel cost of an s-t-path P. We interpret $f_2(P)$ as follows: For $(i,j) \in P$, the term $c(P_{s,j}) + d_c(i,j)$ stands for the cost of the first part $P_{s,j}$ of P plus the cost of traveling from j back to i and further to t without using arc (i,j). The value $f_2(P)$ hence stands for the maximal cost of an alternative path if, traveling on P, the path P is blocked at some vertex $v \in V$ and this blocking was unknown before reaching v. (A blocking at a vertex is the worst that can happen – in this case the whole edge has to be traversed backward.) We call $f_2(P)$ the detour cost of P.

The *shortest paths with shortest detours problem* (SPSDP) is defined as

$$\min(f_1(P),\, f_2(P)) \text{ s.t. } P \in \mathcal{P}_{s,t}(G).$$

SPSDP is a biobjective routing problem with solution set $\mathcal{P}_{s,t}(G)$ and with image set $\mathrm{im}(f_1, f_2) \subset \mathbb{R} \times (\mathbb{R} \cup \{\infty\})$. Since we allow the detour cost $f_2(P)$ to be infinity, a necessary and sufficient condition for the feasibility of SPSDP is the existence of an s-t-path in G.

Since we consider more than one objective function, the optimality concept known from single-objective problems, with only one optimal objective value, is not applicable. Instead, we are interested in the set of efficient solutions $P \in \mathcal{P}_{s,t}(G)$ and the set of nondominated points $(l,u) \in \mathrm{im}(f_1, f_2)$ in the Pareto-sense (e.g., [4]):

Definition 2.1 For two paths $P^1, P^2 \in \mathcal{P}_{s,t}(G)$, we say that the path P^1 *dominates* the path P^2 iff the path P^1 is at least as good as P^2 in both criteria and better in at least one of the criteria, i.e., it holds that $f_k(P^1) \leq f_k(P^2)$ for $k \in \{1,2\}$ and $(f_1(P^1), f_2(P^1)) \neq (f_1(P^2), f_2(P^2))$. If there does not exist an s-t-path that dominates the path $P^1 \in \mathcal{P}_{s,t}(G)$, the path P^1 is called *efficient*. Similarly, we say that a point $(l_1,u_1) \in \mathrm{im}(f_1, f_2)$ in the solution set dominates a point $(l_2,u_2) \in \mathrm{im}(f_1, f_2)$ iff it holds that $l_1 \leq l_2$, $u_1 \leq u_2$ and $(l_1,u_1) \neq (l_2,u_2)$. The point (l_1,u_1) is called *nondominated* iff it is the objective value of an efficient solution $P^1 \in \mathcal{P}_{s,t}(G)$.

Remark 2.1 The SPSDP is strongly connected to the biobjective shortest path problem (BSP) [7,25], which is defined as

$$\min(f_1(P),\, \overline{f_2}(P)) \text{ s.t. } P \in \mathcal{P}_{s,t}(G)$$

with bottleneck function $\overline{f_2}(P) = \max_{(i,j)\in P} d_c(i,j)$ as second criterion.
However, SPSDP differs structurally from BSP: In contrast to $\overline{f_2}$, the second objective function f_2 of SPSDP is given by the maximum over $(i,j) \in P$ of $c(P_{s,j}) + d_c(i,j)$, i.e., the terms of which the maximum is taken depend both on P and on $(i,j) \in P$. Hence, the second objective function of SPSDP considers the cost of a path on a

"global" level taking into account both arc and path costs, whereas the second objective function of BSP operates on a "local" level considering only arc costs.
In order to deal with this global scope, an adapted solution algorithm is required: The example graph in Fig. 1 shows that the minimal complete sets of efficient solutions for BSP and SPSDP may indeed differ. It is easy to see that $\{P_0, P_1, P_2\}$ is a minimal complete set of efficient solutions for BSP, while we will see later on that a minimal complete set of efficient solution for SPSDP is given by $\{P_0, P_2\}$.

In this article, we present an algorithm that returns a minimal complete set of efficient s-t-paths for SPSDP, i.e., a set of paths $\mathcal{P} \subset \mathcal{P}_{s,t}(G)$ that contains precisely one path $P \in \mathcal{P}$ with $(f_1(P), f_2(P)) = (l, u)$ for each nondominated point $(l, u) \in \mathrm{im}(f_1, f_2) \subset \mathbb{R} \times (\mathbb{R} \cup \{\infty\})$. Moreover, we study the structure of the problem.

3 Solution Algorithm

In order to find a minimal complete set of efficient solutions for SPSDP, we use the threshold algorithm presented below, which iteratively deletes arcs that imply expensive detour costs. The idea of the algorithm is based on a solution algorithm for the biobjective shortest path problem with the bottleneck function $\overline{f_2}(P) = \max_{(i,j)\in P} c(i, j)$ as second criterion [7,25]. The structural difference between this biobjective shortest path problem with bottleneck function and the SPSDP is pointed out in Remark 2.1. See [26] for multiobjective bottleneck problems in general.

An example illustrating the key features of the SPSDP algorithm is found in Fig. 1. We choose to include parallel arcs in the example graph in order to keep it comprehensible. The steps in the algorithm are the following:

First, the algorithm initializes the candidate set of efficient paths $\mathcal{P}$ to be empty. Second, the cost l_{-1} of a virtual starting path is set to be $-\infty$ and its detour cost is set to be ∞.

In the k-th iteration, calculate a shortest path P_k in the current version of the graph G, its cost $l_k \leftarrow f_1(P_k)$ and its detour cost $u_k \leftarrow f_2(P_k)$.

If the cost l_k is greater than the minimal detour cost $u_{k'}$ of a previously calculated path $P_{k'}$, i.e., it holds that $-1 \leq k' \leq k$, the algorithm terminates. We will show that, in this case, the current version of the graph does not contain an efficient path for the original problem.

If the detour cost u_k of the path P_k is smaller than the detour cost of all previously calculated paths, we add P_k to the candidate set of efficient paths $\mathcal{P}$. If the cost l_k of P_k is equal to l_{k-1} and if it holds that $P_{k-1} \in \mathcal{P}$, we remove P_{k-1} from the candidate set $\mathcal{P}$. We will see that P_{k-1} is not efficient in this case.

Next, we update the graph by deleting arcs that produce expensive detour costs: For every arc $(i, j) \in A$, we calculate the current estimated detour cost

$$d(i, j) = \left(\min_{P \in \mathcal{P}_{s,j}(G)} c(P) \right) + d_c(i, j).$$

The term $d(i, j)$ is a lower bound on the detour cost of a path (in the current graph) that uses the arc (i, j). We delete all those arcs $(i, j) \in A$ from G whose estimated

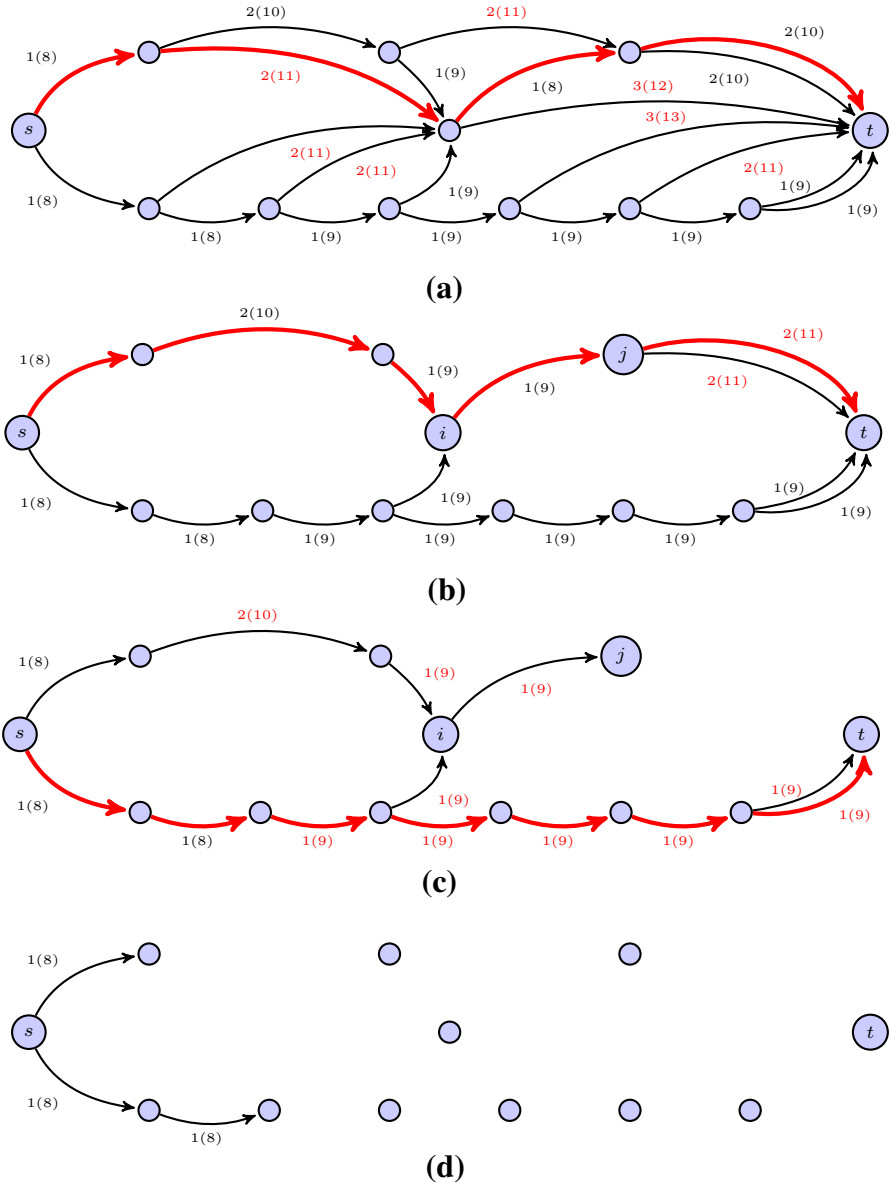

Fig. 1 Example of SPSDP algorithm with turning cost $c' = c$ and detour cost $d_c = d_c^1$, see Example 2.1. For each arc, the arc cost and, in parentheses, the current estimated detour cost are indicated. **a** The *thick red* path P_0 of length $l_0 = 6$ is a shortest path and added to $\mathcal{P} = \{P_0\}$. Delete all arcs with estimated detour cost at least $u_0 = 11$. **b** Now, the *thick red* path $\mathcal{P}_1$ is a shortest s-t-path with $u_1 = u_0$, which is not added to $\mathcal{P}$. Delete all arcs with estimated detour costs at least $u_1 = 11$. *Note*: Some estimated detour costs, e.g., on (i, j), have changed because the length of a shortest s-i-path has changed. **c** The *thick red* path $\mathcal{P}_2$ is a shortest s-t-path with $u_2 = 9 < u_1$ of length $l_2 = 7$ and is, therefore, added to $\mathcal{P}$. Delete all arcs with estimated detour costs at least $u_2 = 9$. **d** No s-t-path exists and the algorithm terminates. Return $\mathcal{P} = \{P_0, P_2\}$

Algorithm 1: SPSDP

Input : A finite directed graph $G = (V, A)$, distinct vertices $s, t \in V$, a cost function $c : A \to \mathbb{R}_{\geq 0}$, a detour cost function $d_c : A \to [0, \infty]$
Output: A minimal complete set $\mathcal{P} \subset \mathcal{P}_{s,t}(G)$ of efficient solutions for SPSDP

1 $\mathcal{P} \leftarrow \emptyset, l_{-1} \leftarrow -\infty, u_{-1} \leftarrow \infty, k \leftarrow 0$
2 **while** $\mathcal{P}_{s,t}(G) \neq \emptyset$ **do**
3 **for** $i \in V$ **do**
4 calculate a shortest s-i-path in G if possible
5 denote the shortest s-t-path calculated in G by P_k, calculate $f_2(P_k)$
6 $l_k \leftarrow f_1(P_k), u_k \leftarrow f_2(P_k)$
7 **if** $l_k > \min_{-1 \leq k' \leq k} u_{k'}$ **then**
  // the length of P_k is larger than the detour cost of all paths
  // computed before
8 **break**
9 **if** $u_k < \min_{-1 \leq k' < k} u_{k'}$ or $k = 0$ **then**
  // $k = 0$ or the detour cost of P_k is smaller than the detour
  // cost of all paths computed before
10 add P_k to $\mathcal{P}$
11 **if** $l_k = l_{k-1}$ **then**
12 **if** $P_{k-1} \in \mathcal{P}$ **then** // $k - 1 \geq 0$ since $l_{-1} = -\infty$
13 remove P_{k-1} from $\mathcal{P}$
14 **for** $(i, j) \in A$ **do**
  // calculate an estimated detour cost of all arcs
15 $d(i, j) \leftarrow \min_{P \in \mathcal{P}_{s,j}(G)} c(P) + d_c(i, j)$
16 **for** $(i, j) \in A$ **do**
17 **if** $d(i, j) \geq \min_{-1 \leq k' \leq k} u_{k'}$ **then**
  // the estimated detour cost of (i, j) is at least
  // the detour cost of all previously calculated paths
18 remove (i, j) from A
19 $k \leftarrow k + 1$
20 **return** $\mathcal{P}$

detour cost $d(i, j)$ is at least the minimal detour cost $\min_{-1 \leq k' \leq k} u_{k'}$ of the previously calculated paths.

The algorithm returns a set $\mathcal{P} \subset \mathcal{P}_{s,t}(G)$ of paths, which will turn out to be a minimal complete set of efficient solutions for SPSDP.

With respect to iteration $k \geq 0$, we denote by

- A_k the set of arcs at the beginning of iteration k,
- $G_k = (V, A_k)$ the graph at the beginning of iteration k,
- P_k the path that is calculated in iteration k,
- $l_k = f_1(P_k)$ the cost and $u_k = f_2(P_k)$ the detour cost of P_k,
- $d^k(i, j)$ the estimated detour cost of arc $(i, j) \in A_k$, as computed in line 15 of iteration k, and
- $\mathcal{P}_{k-1}$ the candidate set of efficient paths at the beginning of iteration k.

For proving termination of the presented algorithm, we need the following lemmata. The first one states a lower and an upper bound on the detour cost.

Lemma 3.1 *Let $P \in \mathcal{P}_{s,t}(G)$ be an s-t-path. Then*:

(a) *The cost of a path is always smaller than or equal to its detour cost, i.e.,*

$$0 \leq f_1(P) \leq f_2(P).$$

(b) *If it holds that $f_2(P) < \infty$, the detour cost of P is bounded from above by*

$$f_2(P) \leq \sum_{(i,j)\in A} c(i,j) + \max_{(i,j)\in A} d_c(i,j).$$

Proof Part (a): Let (i', t) be the last arc on the s-t-path P. The detour cost $f_2(P)$ is given by $\max_{(i,j)\in P} c(P_{s,j}) + d_c(i,j)$. In particular, it holds that

$$f_2(P) \geq c(P_{s,t}) + d_c(i',t) \geq c(P_{s,t}).$$

Part (b): The simple path P cannot use more than all arcs of G. □

The following lemma shows that the calculation of the detour cost of a shortest s-t-path in the graph G_k can be simplified.

Lemma 3.2 *Let $k \geq 0$, and let $P \in \mathcal{P}_{s,t}(G_k)$ be a shortest s-t-path in G_k. Then, it holds that*

$$f_2(P) = \max_{(i,j)\in P} d^k(i,j).$$

Proof Since P is a shortest s-t-path in G_k, the path $P_{s,j}$ is a shortest s-j-path in G_k for all $(i,j) \in P$, and we get

$$f_2(P) = \max_{(i,j)\in P} \left(\left(\min_{P'\in\mathcal{P}_{s,j}(G_k)} c(P') \right) + d_c(i,j) \right) = \max_{(i,j)\in P} d^k(i,j).$$

□

Next, we analyze how the length of a shortest path and its detour cost vary in subsequent iterations. Moreover, we show that the number of arcs in the graph G_k strictly decreases in each iteration.

Lemma 3.3 *Let us assume that the while-loop in the SPSDP algorithm is in iteration $k \geq 0$. Then, it holds that*

(a) $l_{k-1} \leq l_k$,
(b) $u_{k-1} > u_k$ *or* $l_{k-1} < l_k$,
(c) $|A_k| > |A_{k+1}|$, *i.e., the number of arcs in G_k decreases in each iteration.*

Proof It holds that $l_{-1} = -\infty$ and, for $k \geq 0$, l_k is defined as the cost of a shortest path in G_k. It follows that $l_{-1} \leq l_0$. For $k - 1 \geq 0$, the graph G_k is a subgraph of G_{k-1}, which implies that

$$l_{k-1} \leq l_k.$$

Assume that $l_k = l_{k-1}$, which means that P_k is not only a shortest path in G_k, but also in G_{k-1}. Due to $l_{-1} = -\infty$, it follows that $k - 1 \geq 0$. In the construction of A_k from A_{k-1}, precisely those arcs $(i, j) \in A_{k-1}$ are deleted that fulfill

$$d^{k-1}(i, j) \geq \min_{-1 \leq k' \leq k-1} u_{k'}.$$

With Lemma 3.2 and since P_k is a shortest s-t-path in G_{k-1}, we obtain

$$u_k = f_2(P_k) = \max_{(i,j) \in P_k} d^{k-1}(i, j) \leq \max_{(i,j) \in A_k} d^{k-1}(i, j) < \min_{-1 \leq k' \leq k-1} u_{k'} \leq u_{k-1}.$$

Finally, we show $|A_k| < |A_{k+1}|$: It holds that $u_k = \max_{(i,j) \in P_k} d^k(i, j)$ due to Lemma 3.2. Hence, there exists an arc $(i, j) \in P_k$ such that

$$d^k(i, j) = u_k \geq \min_{-1 \leq k' \leq k} u_{k'},$$

and this arc is being removed from A_k in the construction of A_{k+1}. □

Proposition 3.1 *The* SPSDP *algorithm terminates after at most* $|A|$ *iterations in time* $\mathcal{O}\left(|A| \cdot \left(|V| \log |V| + |A|\right)\right)$.

Proof Since the number of arcs in G_k strictly decreases in each iteration, see Lemma 3.3, there exists $0 \leq \tilde{k} \leq |A|$ such that $A_{\tilde{k}} = \emptyset$ and $\mathcal{P}_{s,t}(G_{\tilde{k}}) = \emptyset$. Then, the while-loop breaks, and the SPSDP algorithm returns $\mathcal{P}$.

The most time-consuming step in the while-loop is the shortest path calculation at the beginning, which can be executed in time $\mathcal{O}(|V| \log |V| + |A|)$, see [24], which proves the claim. □

For proving correctness of the SPSDP algorithm, we use the following lemma and proposition. The lemma states that the "first" path $P_{\bar{k}}$ with the lowest detour cost up to iteration $\bar{k} \leq k$ is still contained in the candidate set of efficient paths $\mathcal{P}_k$ in iteration k.

Lemma 3.4 *Let* $k \geq 0$*, and assume that the SPSDP algorithm proceeds to iteration* $k + 1$*. Let* $0 \leq \bar{k} \leq k$ *be the smallest number such that* $u_{\bar{k}} = \min_{-1 \leq k' \leq k} u_{k'}$*. Then, the path* $P_{\bar{k}}$*, which is calculated in iteration* k*, is contained in* P_k*.*

Proof It holds that either $u_{\bar{k}} = \infty$, and therefore $\bar{k} = 0$, or

$$u_{\bar{k}} < \min_{-1 \leq k' < \bar{k}-1} u_{k'}.$$

In both cases $P_{\overline{k}} \in \mathcal{P}_{\overline{k}}$ is added to the candidate set of efficient solutions in iteration $\overline{k}$, see line 9 of the algorithm. If it holds that $\overline{k} = k$, the claim follows. So let us assume that $\overline{k} < k$. Then, it holds that $u_{\overline{k}} \leq u_{\overline{k}+1}$ and hence $l_{\overline{k}} < l_{\overline{k}+1}$, see Lemma 3.3. Therefore, the path $P_{\overline{k}}$ is not being removed from the candidate set of efficient solutions in iteration $\overline{k}+1$, which is the only possibility for removal. □

The following loop invariants characterize the efficient solutions that are contained in the candidate set $\mathcal{P}_{k-1}$ in iteration $k-1$ of the SPSDP algorithm.

Proposition 3.2 (Loop invariants) *Let $k \geq 0$. The following properties are loop invariants for the while-loop in the SPSDP algorithm.*

(a) *For all nondominated points $(l,u) \in \mathrm{im}(f_1, f_2) \subset \mathbb{R} \times (\mathbb{R} \cup \{\infty\})$ with*

$$l \leq l_{k-1} \text{ and } u \geq \min_{-1 \leq k' \leq k-1} u_{k'},$$

the set $\mathcal{P}_{k-1}$ contains exactly one efficient s-t-path $P \in \mathcal{P}_{s,t}(G)$ such that

$$(f_1(P), f_2(P)) = (l,u).$$

The set $\mathcal{P}_{k-1}$ may additionally contain P_{k-1}, but no further elements.

(b) *For all remaining nondominated points $(l,u) \in \mathrm{im}(f_1, f_2)$, there exists an efficient s-t-path $P \in \mathcal{P}_{s,t}(G_k)$ (and not only in $G = G_0$) such that $(f_1(P), f_2(P)) = (l,u)$.*

Proof It is easy to check that the loop invariants are fulfilled for $k = 0$. So let us assume that $k \geq 0$ and that loop invariant (a) is not fulfilled when iteration $k+1$ starts.

I. First, let us assume that there exists a nondominated point $(l,u) \in \mathrm{im}(f_1, f_2)$ such that

$$l \leq l_k \text{ and } u \geq \min_{-1 \leq k' \leq k} u'_k$$

and such that there does not exist a path $P \in \mathcal{P}_k$ with the property $(f_1(P), f_2(P)) = (l,u)$. There are two cases to consider.

i. A path $P \in \mathcal{P}_{k-1}$ fulfills $(f_1(P), f_2(P)) = (l,u)$: It follows that P is removed from the candidate set of efficient solutions in iteration k, hence it holds that $P = P_{k-1}$. This implies $l_k = l_{k-1} = l$, see line 11 of the algorithm, and $u_k < u_{k-1} = u$ with Lemma 3.3, which is a contradiction to (l,u) being nondominated.

ii. $\mathcal{P}_{k-1}$ does not contain a path P that fulfills $(f_1(P), f_2(P)) = (l,u)$: It follows from loop invariant (b) for iteration k that there exists an s-t-path P in G_k that satisfies $(f_1(P), f_2(P)) = (l,u)$. Since l_k is the length of a shortest path in G_k, we conclude $l \geq l_k$ and, due to the assumption $l \leq l_k$, also $l = l_k$.

Let $0 \leq \overline{k} \leq k$ be the smallest number such that $u_{\overline{k}} = \min_{-1 \leq k' \leq k} u_{k'}$. Lemma 3.3 implies that the path $P_{\overline{k}}$, calculated in iteration $\overline{k}$, fulfills $l_{\overline{k}} \leq l_k =$

l. This implies $u \leq u_{\overline{k}}$ because (l, u) is assumed to be nondominated. As we assume $u \geq \min_{-1 \leq k' \leq k} u_{k'}$, we get

$$u = u_{\overline{k}} = \min_{-1 \leq k' \leq k} u'_k.$$

Since (l, u) is assumed to be nondominated, it follows $l = l_{\overline{k}} \leq l_k$. Hence, the path $P_{\overline{k}}$ fulfills $(f_1(P_{\overline{k}}), f_2(P_{\overline{k}})) = (l, u)$, and, according to Lemma 3.4, it holds that $P_{\overline{k}} \in \mathcal{P}_k$. This is a contradiction to the assumption in (i).

II. For each nondominated point $(l, u) \in \mathrm{im}(f_1, f_2)$ fulfilling

$$l \leq l_{k-1} \text{ and } u \geq \min_{-1 \leq k' \leq k-1} u_{k'},$$

the set $\mathcal{P}_{k-1}$ contains at most one efficient path $P \in \mathcal{P}_{s,t}(G)$ such that $(f_1(P), f_2(P)) = (l, u)$: This is true because different paths in $\mathcal{P}$ induce different values in the image $\mathrm{im}(f_1, f_2)$, see Lemma 3.3.

III. We assume next that there exists a path $P' \in \mathcal{P}_k$ that is not efficient and that fulfills $P' \neq P_k$. In particular, it holds that $k \geq 1$. Since the set $\mathcal{P}_k$ is contained in $\mathcal{P}_{k-1} \cup \{P_k\}$, it follows from loop invariant (a) for k that $P' \in \{P_{k-1}, P_k\}$ and, hence, $P' = P_{k-1}$. Moreover, it follows from loop invariant (b) for k that there exists a path $P \in \mathcal{P}_{s,t}(G_k)$ that dominates P_{k-1}. Since $P_{k-1} \in \mathcal{P}_k$ is not removed from $\mathcal{P}_k$ in the k-th iteration and since l_k is the length of a shortest path in G_k, it holds that $l_{k-1} < l_k \leq f_1(P)$. This contradicts the assumption that P dominates P_{k-1}.

Now, let us assume that loop invariant (b) is not fulfilled for $k+1$. Then, there exists a nondominated $(l, u) \in \mathrm{im}(f_1, f_2)$ with $l > l_k$ or $u < \min_{-1 \leq k' \leq k} u_{k'}$ such that there does not exist a path $P \in \mathcal{P}_{s,t}(G_{k+1})$ that fulfills the equality $(f_1(P), f_2(P)) = (l, u)$. With Lemma 3.3, it also holds that

$$u < \min_{-1 \leq k' \leq k-1} u_{k'} \text{ or } l > l_k \geq l_{k-1}.$$

Hence, it follows from loop invariant (b) for k that there exist an s-t-path P in G_k such that $(f_1(P), f_2(P)) = (l, u)$. Since l_k is the length of a shortest path in G_k, we obtain $l_k \leq l$.

Let $\overline{k} \geq 0$ be the smallest number such that $u_{\overline{k}} = \min_{-1 \leq k' \leq k} u_{k'}$. It holds that $(l, u) = (l_{\overline{k}}, u_{\overline{k}})$: Due to Lemma 3.3, the equation $l \geq l_k \geq l_{\overline{k}}$ follows. Moreover, the path $P \in \mathcal{P}_{s,t}(G_k)$ contains an arc $(i, j) \in A_k$ that is not contained in A_{k+1} since otherwise P would be contained in $\mathcal{P}_{s,t}(G_{k+1})$. Therefore, the arc (i, j) has been deleted in iteration k, which implies

$$u \geq \max_{(v,w) \in P} d^k(v, w) \geq d^k(i, j) \geq \min_{-1 \leq k' \leq k} u_{k'} = u_{\overline{k}}.$$

Since (l, u) is assumed to be nondominated, we conclude $(l, u) = (l_{\overline{k}}, u_{\overline{k}})$. This is a contradiction to $l > l_k \geq l_{\overline{k}}$ or $u < \min_{-1 \leq k' \leq k} u_{k'} = u_{\overline{k}}$. □

Theorem 3.1 *The SPSDP algorithm described above returns a minimal complete set* $\mathcal{P} \subset \mathcal{P}_{s,t}(G)$ *of efficient solutions for* SPSDP.

Proof Denote $k \geq 0$ as the iteration in which the while-loop of the SPSDP algorithm breaks, and assume that $\mathcal{P}_k$ is not a minimal complete set of efficient solutions.

I. Assume first that the while-loop breaks because there does not exist any s-t-path in G_k. This means that there are no efficient paths in G_k, and we conclude, with loop invariant (b), that there are no nondominated points $(l, u) \in \mathrm{im}(f_1, f_2)$ that fulfill $l > l_{k-1}$ or $u < \min\limits_{-1 \leq k' \leq k-1} u_{k'}$.

   If it holds that $P_{k-1} \notin \mathcal{P}_k$ or if the path P_{k-1} is efficient, it follows from loop invariant (a) that $\mathcal{P} = \mathcal{P}_{k-1}$ is a minimal complete set of efficient solutions for SPSDP.

   So let us assume that $P_{k-1} \in \mathcal{P}_{k-1}$ and that the path P_{k-1} is not efficient. Then, it holds, due to $P_{k-1} \in \mathcal{P}_{k-1}$, that $u_{k-1} < \min_{-1 \leq k' \leq k-2} u_{k'}$ and that none of the paths $P_0, \ldots, P_{k-2}$ dominate P_{k-1}. It follows from loop invariant (b) that there exists an efficient s-t-path P in G_k that dominates P_{k-1}, which is a contradiction to the assumption that there are no efficient s-t-paths in G_k.

II. Now let us assume that the while-loop breaks because it holds that

$$l_k > \min_{-1 \leq k' \leq k} u_{k'}.$$

   Let $\overline{k}$ be the smallest argument for which this minimum is attained. With Lemma 3.1, the equation

$$f_2(P) \geq f_1(P) \geq l_k > u_{\overline{k}} \geq l_{\overline{k}}$$

   follows for all s-t-paths P in G_k. Hence, P_k dominates all s-t-paths in G_k, and there are no s-t-paths in G_k that are efficient for the original problem. Now, the claim follows with the same argument as in case I. □

Having proven the correctness and termination of the SPSDP algorithm, we state additional results on the structure of SPSDP in the remainder of this section.

Corollary 3.1 *The number of nondominated points for* SPSDP *is bounded by the number of arcs.*

Proof In each iteration at most one path is added to the candidate set of efficient solutions $\mathcal{P}_k$. Consequently, the claim
follows from Theorem 3.1 and Lemma 3.1. □

Remark 3.1 Although the number of nondominated points for SPSDP is bounded by $|A|$, there can be exponentially many efficient solutions, e.g., in the (quadratic) grid graph, in which all arc lengths are equal.

Example 3.1 Let $k \in \mathbb{N}$, and let the detour cost function be given by $d_c = d_c^k$, which is being calculated from $c' : A \to [0, \infty]$ as in Example 2.1. Using the presented algorithm, SPSDP can be solved in time $\mathcal{O}\Big(|A| \cdot \big(|V| \log |V| + |A|\big)\Big)$.
Note that the solution of SPSDP does not include the calculation of d_c^k from the cost function c'. So let us assume that we know c', but not d_c^k. If it holds that $k = 1$, the calculation of the detour cost function $d_c^k = d_c^1$, together with the solution of SPSDP, can still be done in time $\mathcal{O}\Big(|A| \cdot \big(|V| \log |V| + |A|\big)\Big)$. For $k > 1$, this extended problem, which includes both the calculation of d_c^k from c' and the solution of SPSDP, turns PSPACE-complete. Both statements follow from Example 2.1.

Theorem 3.1 says that the number of nondominated points is bounded by $|A|$. Its proof is indirect and based on the correctness and termination of the SPSDP algorithm. On one hand, the following proposition is the key to a direct proof of this statement. On the other hand, the proposition provides more information about the structure of the set of nondominated points.

Proposition 3.3 *Let $P^1, P^2 \in \mathcal{P}_{s,t}(G)$ be two efficient paths and assume that there exists an arc $(i, j) \in P^1 \cap P^2$ such that both paths attain the maximum in the detour function*

$$\begin{aligned} f_2 : \mathcal{P}_{s,t}(G) &\to \mathbb{R} \cup \{\infty\} \\ P &\mapsto \max_{(k,l) \in P} \big(c(P_{s,l}) + d_c(k, l)\big). \end{aligned}$$

at arc (i, j). Then, both paths induce the same nondominated point in the image $\mathrm{im}(f_1, f_2)$*, i.e.,* $(f_1(P^1), f_2(P^1)) = (f_1(P^2), f_2(P^2))$.

Proof We assume that the paths P^1 and P^2 are efficient and that it holds $f_1(P^1) \neq f_1(P^2)$ or $f_2(P^1) \neq f_2(P^2)$. We split the two paths into their first part $P_{s,i}^k$ and their second part $P_{j,t}^k$, $k \in \{1, 2\}$. By symmetry, only two cases occur:

I. $c\left(P_{s,i}^1\right) \geq c\left(P_{s,i}^2\right)$ and $c\left(P_{j,t}^1\right) \geq c\left(P_{j,t}^2\right)$:

The cost $f_1(P^1)$ is at least the cost of P^2. Since the maximum in the detour function f_2 is attained at $(i, j) \in A$ for both paths, we get

$$f_2(P^1) = c\left(P_{s,i}^1\right) + c(i, j) + d_c(i, j) \geq c\left(P_{s,i}^2\right) + c(i, j) + d_c(i, j) = f_2(P^2).$$

This means that P^2 dominates P^1, which is a contradiction.

II. $c\left(P_{s,i}^1\right) \leq c\left(P_{s,i}^2\right)$ and $c\left(P_{j,t}^1\right) \geq c\left(P_{j,t}^2\right)$:

We consider a third path

$$P^3 = P_{s,i}^1 \circ (i, j) \circ P_{j,t}^2,$$

which fulfills that $f_1(P^3) \leq f_1(P^k)$, where $k \in \{1, 2\}$. We claim that P^3 attains the maximum in the detour function f_2 at the arc (i, j):
Assume that there exists an arc $(i, j) \neq (v, w) \in P^3$ which fulfills that

$$c\left(P^3_{s,w}\right) + d_c(v, w) > c\left(P^3_{s,j}\right) + d_c(i, j).$$

If the arc (v, w) lies in front of the arc (i, j) on P^3, it follows that

$$c\left(P^3_{s,w}\right) = c\left(P^1_{s,w}\right) \text{ and } c\left(P^3_{s,j}\right) = c\left(P^1_{s,j}\right).$$

Furthermore, the maximum in the detour function $f_2(P^1)$ is not attained at (i, j) either.

So assume that (v, w) lies behind (i, j) on P^3. For all vertices w' that lie behind the vertex i on P^3, it holds that

$$c\left(P^2_{s,w'}\right) = c\left(P^3_{s,w'}\right) + \left(c\left(P^2_{s,i}\right) - c\left(P^1_{s,i}\right)\right),$$

i.e., the travel costs of $P^2_{s,w'}$ and $P^3_{s,w'}$ differ only by a constant. It follows that

$$c\left(P^2_{s,w}\right) + d_c(v, w) > c\left(P^2_{s,j}\right) + d_c(i, j),$$

and the maximum in the detour function f_2 of P^2 is not attained at (i, j).

Hence, P^1, P^2 and P^3 obtain the maximum in the detour function at (i, j), and it follows that

$$\begin{aligned} f_2(P^3) = f_2(P^1) &= c\left(P^1_{s,i}\right) + c(i, j) + d_c(i, j) \\ &\leq c\left(P^2_{s,i}\right) + c(i, j) + d_c(i, j) = f_2(P^2). \end{aligned}$$

We conclude that P^3 is at least as good as P^1 and P^2 in both criteria f_1 and f_2. This is a contradiction to the assumption that P^1 and P^2 are efficient, but correspond to different nondominated points in the image $\mathrm{im}(f_1, f_2)$. □

4 Conclusions

In this article, we introduce the biobjective routing problem SPSDP of finding shortest paths with shortest detours and present an algorithm that solves SPSDP in polynomial time. Moreover, we prove several structural statements. In particular, we show that the number of nondominated points for SPSDP is bounded by the number of arcs in G.

We studied the concept of arc failures in graphs. One could carry this idea and the SPSDP algorithm over to combinatorial optimization and study the class of problems that are solvable with it. Another future research idea is to apply the concept of blockings to multiobjective shortest paths with several arc costs $c_k : A \to \mathbb{R}_{\geq 0}$, $k = 1, \ldots, q$. Each cost function c_k implies a detour cost function $d_{c_k} : A \to \mathbb{R}$ on the arcs and hence also on the set of s-t-paths. One could, on one hand, scalarize these detour functions to one objective function measuring the recovery cost. On the other hand, it would be interesting to study the arising multiobjective problem with one detour cost function for each arc cost c_k.

Acknowledgements The authors are partly supported by the German Federal Ministry of Education and Research (BMBF), Reference Number 13N12825.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Upper Saddle River (1993)
2. Demetrescu, C., Goldberg, A.V., Johnson, D.S. (eds.): The Shortest Path Problem. Series in Discrete Mathematics and Computer Science, vol. 74. American Mathematical Society, Providence (2009)
3. Martins, E.Q.V.: On a multicriteria shortest path problem. Eur. J. Oper. Res. **16**, 236–245 (1984)
4. Ehrgott, M.: Multicriteria Optimization. Lecture Notes in Economics and Mathematical Systems, vol. 491, 2nd edn. Springer, Berlin (2005)
5. Ulungu, E., Teghem, J.: Multi-objective shortest path problem: a survey. In: Proceedings of the International Workshop on Multicriteria Decision Making: Methods–Algorithms–Applications at Liblice, Czechoslovakia, pp. 176–188 (1991)
6. Serafini, P.: Some considerations about computational complexity for multi objective combinatorial problems. In: Jahn, J., Krabs, W. (eds.) Recent Advances and Historical Development of Vector Optimization. Lecture Notes in Economics and Mathematical Systems, vol. 294, pp. 222–232. Springer, Berlin (1986)
7. Hansen, P.: Bicriterion path problems. In: Fandel, G., Gal, T. (eds.) Lecture Notes in Economics and Mathematical Systems, vol. 177, pp. 109–127. Springer, Berlin (1980)
8. Goerigk, M., Schöbel, A.: Algorithm engineering in robust optimization. In: Kliemann, L., Sanders, P. (eds.) Algorithm Engineering, pp. 245–279. Springer, Cham, Switzerland (2016)
9. Kouvelis, P., Yu, G.: Robust Discrete Optimization and Its Applications. Nonconvex Optimization and Its Applications. Springer, Dordrecht (1997)
10. Yu, G., Yang, J.: On the robust shortest path problem. Comput. Oper. Res. **25**(6), 457–468 (1998)
11. Zielinski, P.: The computational complexity of the relative robust shortest path problem with interval data. Eur. J. Oper. Res. **158**, 570–576 (2004)
12. Averbakh, I., Lebedev, V.: Interval data minmax regret network optimization problems. Discrete Appl. Math. **138**, 289–301 (2004)
13. Nardelli, E., Proietti, G., Widmayer, P.: Finding the detour-critical edge of a shortest path between two nodes. Inf. Process. Lett. **67**(1), 51–54 (1998)
14. Xu, Y., Yan, H.: Real time critical edge of the shortest path in transportation networks. In: Cai, J.Y., Cooper, S., Li, A. (eds.) Theory and Applications of Models of Computation. Lecture Notes in Computer Science, vol. 3959, pp. 198–205. Springer, Berlin (2006)
15. Hershberger, J., Suri, S.: Vickrey prices and shortest paths: What is an edge worth? In: Proceedings of 42nd IEEE Symposium on Foundations of Computer Science, 2001, pp. 252–259. IEEE (2001)
16. Corley, H., David, Y.S.: Most vital links and nodes in weighted networks. Oper. Res. Lett. **1**(4), 157–160 (1982)
17. Nardelli, E., Proietti, G., Widmayer, P.: Finding the most vital node of a shortest path. Theor. Comput. Sci. **296**, 167–177 (2003)
18. Malik, K., Mittal, A., Gupta, S.: The k most vital arcs in the shortest path problem. Oper. Res. Lett. **8**(4), 223–227 (1989)
19. Israeli, E., Wood, R.K.: Shortest-path network interdiction. Networks **40**(2), 97–111 (2002)

20. Rocco, C.M., Ramirez-Marquez, J.E.: A bi-objective approach for shortest-path network interdiction. Comput. Indus. Eng. **59**(2), 232–240 (2010)
21. Liebchen, C., Lübbecke, M., Möhring, R., Stiller, S.: The concept of recoverable robustness, linear programming recovery, and railway applications. In: Ahuja, R.K., Möhring, R.H., Zaroliagis, C.D. (eds.) Robust and Online Large-Scale Optimization, vol. 5868. Springer, Berlin (2009)
22. Papadimitriou, C.H., Yannakakis, M.: Shortest paths without a map. In: Ausiello, G., Dezani-Ciancaglini, M., Della Rocca, S.R. (eds.) Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 372. Springer, Berlin (1989)
23. Bar-Noy, A., Schieber, B.: The Canadian traveller problem. In: Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms (1991)
24. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM **34**(3), 596–615 (1987)
25. Pelegrin, B., Fernandez, P.: On the sum-max bicriterion path problem. Comput. Oper. Res. **25**(12), 1043–1054 (1998)
26. Gorski, J., Klamroth, K., Ruzika, S.: Generalized multiple objective bottleneck problems. Oper. Res. Lett. **40**(4), 276–281 (2012)