Multi-objective minmax robust combinatorial optimization with cardinality-constrained uncertainty

Andrea Raith[a], Marie Schmidt[b], Anita Schöbel[c], Lisa Thom[c,*]

[a] *Department of Engineering Science, The University of Auckland, Private Bag 92019, Auckland 1142, New Zealand*
[b] *Department of Technology and Operations Management, Rotterdam School of Management, Erasmus University Rotterdam, PO Box 1738, 3000 DR Rotterdam, The Netherlands*
[c] *Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen, Lotzestr. 16-18, 37083 Göttingen, Germany*

Abstract

In this paper we develop two approaches to find minmax robust efficient solutions for multi-objective combinatorial optimization problems with cardinality-constrained uncertainty. First, we extend an existing algorithm for the single-objective problem to multi-objective optimization. We propose also an enhancement to accelerate the algorithm, even for the single-objective case, and we develop a faster version for special multi-objective instances. Second, we introduce a deterministic multi-objective problem with sum and bottleneck functions, which provides a superset of the robust efficient solutions. Based on this, we develop a label setting algorithm to solve the multi-objective uncertain shortest path problem. We compare both approaches on instances of the multi-objective uncertain shortest path problem originating from hazardous material transportation.

Keywords: Multiple objective programming, Robust optimization, Combinatorial optimization, Multi-objective robust optimization, Shortest path problem

1. Introduction

Two of the main difficulties in applying optimization techniques to real-world problems are that several (conflicting) objectives may exist and that parameters may not be known exactly in advance. In multi-objective optimization several objectives are optimized simultaneously by choosing solutions that cannot be improved in one objective without worsening it in another objective. Robust optimization hedges against (all) possible parameter values, e.g., by assuming the worst case for each solution (minmax robustness).
Often it is assumed that the uncertain parameters take any value from a given interval or that discrete scenarios are given. A survey on robust combinatorial optimization with these uncertainty sets is given by Aissi et al. (2009). Based on the interval case, Bertsimas and Sim (2004) propose to consider scenarios where only a bounded number of parameters differ from their expected value (cardinality-constrained uncertainty). This leads to less conservative solutions that are of high practical use. Bertsimas and Sim (2003) provide an algorithm to

*Corresponding author
Email addresses: a.raith@auckland.ac.nz (Andrea Raith), schmidt2@rsm.nl (Marie Schmidt), schoebel@math.uni-goettingen.de (Anita Schöbel), l.thom@math.uni-goettingen.de (Lisa Thom)

find robust solutions for combinatorial optimization problems under this kind of uncertainty. Only recently have robust optimization concepts for multi-objective problems been developed. Kuroiwa and Lee (2012) and Fliege and Werner (2014) introduce a first extension of minmax robustness for several objectives. They consider the uncertainties in the objectives independently of each other. Ehrgott et al. (2014) develop another extension of minmax robustness, in which they include the dependencies between the objectives. This is further generalized by Ide et al. (2014). These concepts have been extensively applied, e.g., in portfolio management (Fliege and Werner, 2014), in game theory (Yu and Liu, 2013) and in the wood industry (Ide et al., 2015). Ide and Schöbel (2016) and Wiecek and Dranichak (2016) give an overview on multi-objective robustness, including further robustness concepts. Newest developments in this field include works by Chuong (2016) and Kalantari et al. (2016). Cardinality constrained uncertainty is extended to multi-objective optimization by Doolittle et al. (2012) (only for uncertain constraints) and Hassanzadeh et al. (2013) (for uncertain objective functions and constraints).
To the best of our knowledge, only Kuhn et al. (2016) have developed a solution algorithm for multi-objective uncertain combinatorial optimization problems. They consider problems with two objectives, of which only one is uncertain, with discrete and polyhedral uncertainty sets.
In this paper, however, we consider problems with any fixed number of objectives of which all may be uncertain. The main contribution of this paper is that we develop two solution approaches for multi-objective combinatorial optimization problems with cardinality-constrained uncertainty. We further derive specific algorithms for the multi-objective uncertain shortest path problem.
The remainder of this paper is structured as follows: In Section 2 we give a short introduction to multi-objective robust optimization. We present two solution approaches for multi-objective combinatorial optimization problems with cardinality-constrained uncertainty in Section 3: In Section 3.1 we extend an algorithm by Bertsimas and Sim (2003) to multi-objective optimization. Additionally, we propose an acceleration for both the single-objective and the multi-objective case and a faster version for multi-objective problems with a special property. In Section 3.2 we introduce a second approach and show how it can be applied to solve the multi-objective uncertain shortest path problem as an example. In Section 4, we compare our methods on instances of the multi-objective uncertain shortest path problem originating from hazardous material transportation.

2. Multi-objective combinatorial optimization with cardinality-constrained uncertainty

First, we give an introduction to multi-objective combinatorial optimization. We use bold font for vectors and vector valued functions.
An instance $(E, Q, \boldsymbol{c})$ of a multi-objective combinatorial optimization problem is given by a finite element set E, a set $Q \subseteq 2^E$ of feasible solutions, which are subsets of E, and a *cost function* $\boldsymbol{c}$, that assigns a *cost vector* $\boldsymbol{c}(e) = (c_1(e), ..., c_k(e))$ to each element $e \in E$. The *cost* $\boldsymbol{z}(q)$ of a set $q \in Q$ is the sum of the costs of its elements. We call

$$(MOCO) \qquad \min_{q \in Q} \left(\boldsymbol{z}(q) = \begin{pmatrix} z_1(e) \\ \vdots \\ z_k(e) \end{pmatrix} = \begin{pmatrix} \sum_{e \in q} c_1(e) \\ \vdots \\ \sum_{e \in q} c_k(e) \end{pmatrix} \right)$$

a *multi-objective combinatorial optimization problem.*
A solution that minimizes all objectives simultaneously does usually not exist. Therefore, we use the well-known concept of *efficient solutions.*

Notation 1. *For two vectors $\boldsymbol{y}^1, \boldsymbol{y}^2 \in \mathbb{R}^k$ we use the notation*

$$\boldsymbol{y}^1 \leq \boldsymbol{y}^2 \Leftrightarrow y_i^1 \leqq y_i^2 \text{ for } i = 1, ..., k \text{ and } \boldsymbol{y}^1 \neq \boldsymbol{y}^2,$$
$$\boldsymbol{y}^1 \leqq \boldsymbol{y}^2 \Leftrightarrow y_i^1 \leqq y_i^2 \text{ for } i = 1, ..., k.$$

In the following, we only use the symbols $<$ (strictly less than) and $\leqq$ (less than or equal to) to compare scalars.

Definition 2. *A solution $q' \in Q$* dominates *another solution $q \in Q$ if $\boldsymbol{z}(q') \leq \boldsymbol{z}(q)$. We also say that $\boldsymbol{z}(q')$ dominates $\boldsymbol{z}(q)$. A solution $q \in Q$ is an* efficient *solution, if there is no $q' \in Q$ such that q' dominates q. Then $\boldsymbol{z}(q)$ is called* non-dominated*.*
Two efficient solutions $q, q' \in Q$ are called equivalent *if $\boldsymbol{z}(q) = \boldsymbol{z}(q')$. A set of efficient solutions $\bar{Q} \subseteq Q$ is called* complete *if all $q \in Q \setminus \bar{Q}$ are either dominated by or equivalent to at least one $q' \in \bar{Q}$.*

Solving (MOCO) means to find a complete set of efficient solutions.

We now assume that the input data is uncertain, i.e., the feasible set and/or the element costs $\boldsymbol{c}(e)$ are not exactly known in advance. If the set of feasible solutions is uncertain, we aim to find solutions which are feasible in all scenarios (as proposed in the seminal works on robustness, see, e.g., Soyster (1973); Ben-Tal et al. (2009)). For this purpose, the sets of feasible solutions can be intersected in advance to obtain a (deterministic) set of *robust feasible solutions.* Hence, in the following, we assume the set Q to be a deterministic set.
The *uncertainty set $\mathcal{U}$* is then the set of all possible cost functions $\boldsymbol{c}$. The considered uncertainty set often strongly influences the solvability of uncertain optimization problems and the solution approaches. The idea of *cardinality-constrained uncertainty* is to assume that the parameters vary in intervals independent of each other, but not more than a given number of elements will be more expensive than their minimal cost. For example, there will not be an accident on every road of a transportation network at the same time, thus, a delay because of an accident does not need to be considered on all roads simultaneously. Bertsimas and Sim (2003) were the first to introduce cardinality-constrained uncertainty for single-objective uncertain combinatorial optimization problems. With $\hat{c}_e$ being the minimal or *nominal* value of $c(e)$ and $\hat{c}_e + \delta_e$ its maximal value, the considered uncertainty set can be written as

$$\mathcal{U}^{cc} := \{c : E \to \mathbb{R} \mid c(e) \in [\hat{c}_e, \hat{c}_e + \delta_e] \ \forall \ e \in E, |\{e \in E \mid c(e) > \hat{c}_e\}| \leqq \Gamma\}. \tag{1}$$

One possible extension to multi-objective optimization is to apply this approach to each objective independently (see Hassanzadeh et al., 2013):

Definition 3. *For each element $e \in E$ and each objective i let two real values $\hat{c}_{e,i}$ and $\delta_{e,i} \geqq 0$ be given. We assume that the uncertain cost $c_i(e)$ can take any value in the interval $[\hat{c}_{e,i}, \hat{c}_{e,i} + \delta_{e,i}]$, with $\hat{c}_{e,i}$ being the undisturbed value, called the* nominal *value. For each objective i let an integer $\Gamma_i \leqq |E|$ be given. The* cardinality-constrained uncertainty set *contains all cost functions, with which for each objective i at most Γ_i elements differ from their nominal costs:*

$$\begin{aligned}\mathcal{U}^{mcc} := \{\boldsymbol{c} : E \to \mathbb{R}^k \ |c_i(e) \in [\hat{c}_{e,i}, \hat{c}_{e,i} + \delta_{e,i}] \ \forall \ e \in E, \ \forall \ i = 1, ..., k, \\ |\{e \in E \mid c_i(e) > \hat{c}_{e,i}\}| \leqq \Gamma_i \ \forall \ i = 1, ..., k\}\end{aligned} \tag{2}$$

We call the family of optimization problems

$$(MOUCO) \qquad \left(\min_{q \in Q} \left(\boldsymbol{z}(q) = \sum_{e \in q} \boldsymbol{c}(e) \right), \boldsymbol{c} \in \mathcal{U}^{mcc} \right)$$

a *multi-objective uncertain combinatorial optimization problem with cardinality-constrained uncertainty.* An instance of (MOUCO) is hence given by $(E, Q, \hat{C}, \Delta, \boldsymbol{\Gamma})$, with

$$\hat{C} := \begin{pmatrix} \hat{c}_{e_1,1} & \dots & \hat{c}_{e_1,k} \\ \vdots & & \vdots \\ \hat{c}_{e_{|E|},1} & \dots & \hat{c}_{e_{|E|},k} \end{pmatrix}, \Delta := \begin{pmatrix} \delta_{e_1,1} & \dots & \delta_{e_1,k} \\ \vdots & & \vdots \\ \delta_{e_{|E|},1} & \dots & \delta_{e_{|E|},k} \end{pmatrix}, \boldsymbol{\Gamma} := (\Gamma_1, ..., \Gamma_k).$$

Note that with the uncertainty set $\mathcal{U}^{mcc}$, (MOUCO) is *objective-wise uncertain*, as it was defined by Ehrgott et al. (2014), i.e., the uncertainty sets in the objective functions are independent of each other.
This can usually be assumed, if the objectives are uncorrelated. However, also for correlated nominal values, the uncertainty can often be assumed to be uncorrelated, if unexpected events influence only one of the objectives.
To decide what is a good solution for a multi-objective uncertain problem is not trivial. In single-objective robust optimization one looks for so-called *robust optimal* solutions. Often these are defined as solutions, which have a minimal worst case value, i.e., one solves $\min_{q \in Q} \max_{c \in \mathcal{U}} z(q)$ (see, e.g., Ben-Tal et al., 2009). This concept has been generalized to robust efficiency for multi-objective problems in various ways (see, e.g., Kuroiwa and Lee, 2012; Ehrgott et al., 2014). In this paper we determine the worst case independently for each objective (see Definition 4), as proposed by Kuroiwa and Lee (2012). This yields a single vector for each solution and these vectors can be compared using the methods of multi-objective optimization.

Definition 4. *A solution $q \in Q$ is* robust efficient *for (MOUCO) if q is an efficient solution for the robust counterpart*

$$(MORCO) \qquad \min_{q \in Q} \left(\boldsymbol{z}^{\boldsymbol{R}}(q) = \begin{pmatrix} \sup\limits_{\boldsymbol{c} \in \mathcal{U}^{mcc}} z_1(q) \\ \vdots \\ \sup\limits_{\boldsymbol{c} \in \mathcal{U}^{mcc}} z_k(q) \end{pmatrix} = \begin{pmatrix} \sup\limits_{\boldsymbol{c} \in \mathcal{U}^{mcc}} \sum\limits_{e \in q} c_1(e) \\ \vdots \\ \sup\limits_{\boldsymbol{c} \in \mathcal{U}^{mcc}} \sum\limits_{e \in q} c_k(e) \end{pmatrix} \right).$$

Remark 5. *Since (MOUCO) is objective-wise uncertain, robust efficiency, as defined in Definition 4, coincides with point-based and set-based minmax robust efficiency defined by Ehrgott et al. (2014). Therefore, all results shown in this paper are valid for both concepts.*

Analogously to Definition 2 we define:

Definition 6. *Two robust efficient solutions $q, q' \in Q$ are called* equivalent *if $\boldsymbol{z}^{\boldsymbol{R}}(q) = \boldsymbol{z}^{\boldsymbol{R}}(q')$. A set of robust efficient solutions $\bar{Q} \subseteq Q$ is called* complete *if all $q \in Q \backslash \bar{Q}$ are either dominated w.r.t. $\boldsymbol{z}^{\boldsymbol{R}}$ or equivalent to at least one $q' \in \bar{Q}$.*

2.1. Example: A multi-objective uncertain shortest path problem

Consider a graph $G = (V, E)$ with node set V and edge set E, a start node $s \in V$ and a destination node $t \in V$. A *path* is a sequence of edges connecting adjacent nodes. In a *simple path* at most two edges are incident to each node. For a given cost function $\boldsymbol{c} : E \to \mathbb{R}^k$ the cost of a path is obtained by following the path and adding up the costs of the edges traversed. Because simple paths do not contain any edge more than once, for a simple path q we have $\boldsymbol{z}(q) = \sum_{e \in q} \boldsymbol{c}(e)$.
In the following we assume *conservative* edge costs, i.e., every cycle C has non-negative cost $z_i(C) \geqq 0$ for each cost function $\boldsymbol{c} \in \mathcal{U}^{mcc}$ and objective $i = 1, ..., k$. Then, there always exists a complete set of robust efficient paths containing only simple paths. Hence, the *multi-objective shortest path problem with cardinality-constrained uncertainty* can be written as a combinatorial problem

$$(MOUSP) \qquad \left(\min_{q \in Q} \sum_{e \in q} \boldsymbol{c}(e), \boldsymbol{c} \in \mathcal{U}^{mcc} \right)$$

with Q being the set of simple paths from s to t in G. We use the following example to illustrate the results and algorithms in this paper.

Example 7. *Consider the network in Figure 1 with $s = v_1$ and $t = v_6$ and $\Gamma_1 = \Gamma_2 = 2$. The edge costs are given in the form*

$$\begin{pmatrix} [\hat{c}_{e,1}, \hat{c}_{e,1} + \delta_{e,1}] \\ [\hat{c}_{e,2}, \hat{c}_{e,2} + \delta_{e,2}] \end{pmatrix}.$$

For this instance of (MOUSP), the set of robust efficient paths consists of the two paths

$$q_1 := \{(v_1, v_2), (v_2, v_4), (v_4, v_6)\} \qquad \text{with } \boldsymbol{z^R}(q_1) = \begin{pmatrix} 13 \\ 9 \end{pmatrix},$$

$$q_2 := \{(v_1, v_2), (v_2, v_3), (v_3, v_5), (v_5, v_6)\} \qquad \text{with } \boldsymbol{z^R}(q_2) = \begin{pmatrix} 11 \\ 16 \end{pmatrix}.$$

3. Algorithms for finding robust efficient solutions in multi-objective uncertain combinatorial optimization

We now consider (MOUCO), hence, we aim to find a complete set of efficient solutions for the robust counterpart (MORCO).

3.1. Deterministic Subproblems Algorithm (DSA)

The algorithms in this subsection are built upon an algorithm by Bertsimas and Sim (2003) for single-objective cardinality-constrained uncertain combinatorial optimization problems, which we call Deterministic Subproblems Algorithm (DSA). Its idea is to find solutions for the uncertain problem by solving up to $|E| + 1$ deterministic problems of the same type and comparing their solutions.

In Section 3.1.2, we first describe the algorithm by Bertsimas and Sim (2003) for single-objective problems. While the authors prove correctness of the algorithm with help of duality, we provide an alternative explanation, which we later extend to (MORCO). In Section 3.1.2,

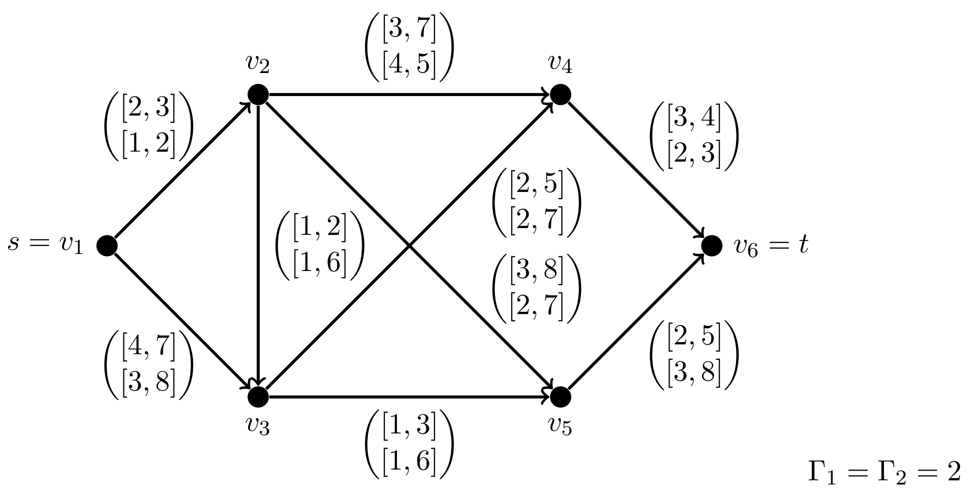


Figure 1: An instance for (MOUSP).

we extend the algorithm for the general multi-objective case and show that the number of subproblems can be further reduced for multi-objective problems with a special property. We present several ways to reduce the number of subproblems to be solved for both the single-objective and the multi-objective case.

3.1.1. The DSA for single-objective problems

We first consider the single-objective problem ($\min_{q \in Q} z(q), c \in \mathcal{U}^{cc}$) with $\mathcal{U}^{cc}$ defined as in Equation (1).

We now explain the algorithm by Bertsimas and Sim (2003). A solution $q \in Q$ has maximal cost (we call this its *worst case cost*), if the costs of those Γ elements, which have the largest cost intervals δ_e among all elements in q, take their maximal values $c(e) = \hat{c}_e + \delta_e$. If q has fewer than Γ elements, in the worst case the cost of all elements in q take their maximal value. Assume that the elements are ordered with respect to the interval length δ, i.e.,

$$\bar{\delta}_1 := \delta_{e_1} \geqq \bar{\delta}_2 := \delta_{e_2} \geqq ... \geqq \bar{\delta}_{|E|} := \delta_{e_{|E|}} \geqq \bar{\delta}_{|E|+1} := 0.$$

For each $l \in \{1, ..., |E| + 1\}$ we define the function g^l (see Bertsimas and Sim, 2003):

$$g^l(q) := \sum_{e \in q} \hat{c}_e + \Gamma \cdot \bar{\delta}_l + \sum_{\substack{e_j \in q \\ j \leqq l}} (\delta_{e_j} - \bar{\delta}_l).$$

The function $g^l(q)$ is an approximation of the worst case costs of the set q. It contains

- the nominal cost $\hat{c}_e$ for each element $e \in q$, which has to be paid also in the worst case,
- $\bar{\delta}_l \cdot \Gamma$ since, in the worst case, the interval length δ_e has to be added to the costs for (at most) Γ elements,
- the positive summand $\max\{0, \delta_e - \bar{\delta}_l\}$ for each element $e \in q$ to account for all elements in the set with higher interval lengths than $\bar{\delta}_l$.

The idea of the algorithm by Bertsimas and Sim (2003) is to solve all problems

$$(\mathcal{P}(l)) \qquad \min_{q \in Q} g^l(q)$$

for $l = 1, \dots |E| + 1$ and compare the worst case values of all obtained solutions to choose a solution with minimal worst case cost. Instead of computing the worst case cost vectors, it is even sufficient to compare the objective values $g^l(q)$ of the obtained solutions and choose the solution with minimal objective value. This idea works due to the following two properties:

1. For every set q and every $l \in \{1, \dots, |E| + 1\}$ we have that $g^l(q)$ is always greater than or equal to the worst case cost $z^{\mathrm{R}}(q)$.
2. For every set q there exists some $l \in \{1, \dots, |E| + 1\}$ such that $g^l(q)$ equals the worst case cost $z^{\mathrm{R}}(q)$.

To show the first property, let q be a set and let $\{e_{a_1}, \dots, e_{a_h}\}$ be a subset of h elements in q with the largest cost intervals, where $h = \min\{|q|, \Gamma\}$. Then $z^{\mathrm{R}}(q) = \sum_{e \in q} \hat{c}_e + \sum_{j=1}^{h} \delta_{e_{a_j}}$ and we get

$$g^l(q) \geqq \sum_{e \in q} \hat{c}_e + \sum_{j=1}^{h} \bar{\delta}_l + \sum_{j=1}^{h} \max\{0, \delta_{e_{a_j}} - \bar{\delta}_l\} \geqq z^{\mathrm{R}}(q).$$

For the second property we show that for each set q there exists at least one index l with $g^l(q) = z^{\mathrm{R}}(q)$: If q has less than Γ elements, then

$$g^{|E|+1}(q) = \sum_{e \in q} \hat{c}_e + \Gamma \cdot 0 + \sum_{e \in q} (\delta_e - 0) = z^{\mathrm{R}}(q).$$

If q has at least Γ elements, let $e_{\bar{l}}$ be the element in q with the Γ-th smallest index. Then the Γ elements $\{e_j \in q : j \leqq \bar{l}\}$ have the largest cost intervals in q and it follows that

$$g^{\bar{l}}(q) = \sum_{e \in q} \hat{c}_e + \Gamma \cdot \bar{\delta}_{\bar{l}} + \sum_{\substack{e_j \in q \\ j \leqq \bar{l}}} (\delta_{e_j} - \bar{\delta}_{\bar{l}}) = \sum_{e \in q} \hat{c}_e + \sum_{\substack{e_j \in q \\ j \leqq \bar{l}}} \bar{\delta}_{\bar{l}} + \sum_{\substack{e_j \in q \\ j \leqq \bar{l}}} (\delta_{e_j} - \bar{\delta}_{\bar{l}}) = z^{\mathrm{R}}(q).$$

Having these two properties, we see that a robust optimal solution q^* is optimal for the problem $(\mathcal{P}(\bar{l}))$, since none of the other sets $q \in Q$ can have a better objective value. Therefore, at least one robust optimal solution will be found by the algorithm.
Algorithm 1 shows the basic structure of the described algorithm. First, the elements are ordered with respect to their interval lengths. Then the subproblems defined above are solved. Finally, of all obtained solutions the one with minimal objective value w.r.t. the respective subproblem is chosen. The efficiency of Algorithm 1 depends on the time complexity to solve the subproblems $(\mathcal{P}(l))$. Because the summand $\Gamma \cdot \bar{\delta}_l$ is solution-independent, a solution for $(\mathcal{P}(l))$ can be found efficiently by solving a problem of the same kind as the underlying deterministic problem with element costs

$$c^l(e_j) := \begin{cases} \hat{c}_{e_j} + (\delta_{e_j} - \bar{\delta}_l) & \text{for } j < l \\ \hat{c}_{e_j} & \text{for } j \geqq l. \end{cases} \tag{3}$$

Hence, Algorithm 1 finds a robust optimal solution in polynomial time for many combinatorial optimization problems. Examples are the minimum spanning tree and the shortest path

Algorithm 1 Basic structure of DSA (based on Bertsimas and Sim, 2003)

```
Input: an instance $I = (E, Q, \hat{\boldsymbol{c}}, \boldsymbol{\delta}, \Gamma)$ of (MOUCO) with $k = 1$
Output: a robust efficient solution for $I$
1: Sort $E$ w.r.t. $\delta_e$ such that $\bar{\delta}_1 := \delta_{e_1} \geqq \bar{\delta}_2 := \delta_{e_2} \geqq ... \geqq \bar{\delta}_{|E|} \geqq \bar{\delta}_{|E|+1} := 0$.
2: Set $L := \{1, ..., |E| + 1\}$.
3: For all $l \in L$ find an optimal solution $q^l$ for $(\mathcal{P}(l))$.
4: Compare the objective values $z^{\mathrm{R}}(q^l)$ for all $l \in L$. The solution with the smallest objective
   value is a robust optimal solution.
```

problem.
In the following, we show how Algorithm 1 can be enhanced. It is not necessary to solve all of the $|E| + 1$ subproblems introduced above. The following three results (see Bertsimas and Sim, 2003; Lee and Kwon, 2014; Park and Lee, 2007) can be used to reduce the number of subproblems (Lemma 8): First, if two elements have the same interval length δ_e, then their associated subproblems are identical. Second, the worst case cost of a set q with at least Γ elements equals its objective value $g^l(q)$ not only for one subproblem, but for two consecutive subproblems. Therefore, we do not miss any solutions if we only solve every second problem. Third, none of the first $\Gamma - 1$ elements can be the one with the Γ-th smallest index for any set in Q, so their associated subproblems need not to be solved.

Lemma 8 ((Bertsimas and Sim, 2003; Lee and Kwon, 2014; Park and Lee, 2007))**.** *The number of subproblems to be solved by Algorithm 1 can be reduced to at most* $\left\lceil \frac{|E|-\Gamma}{2} \right\rceil + 1$ *in the following ways:*

1. *If there are several elements $e_l, ..., e_{(l+h)}$ with the same interval length $\delta_{e_l} = ... = \delta_{e_{l+h}}$, only one of the subproblems $\mathcal{P}(l), ..., \mathcal{P}(l+h)$ needs to be solved (Bertsimas and Sim, 2003).*
2. *Only every second subproblem and the last subproblem need to be solved (Lee and Kwon, 2014).*
3. *It is sufficient to start with the Γ-th subproblem (Park and Lee, 2007).*

Depending on the solutions that are found while the algorithm is executed, we can further reduce the number of subproblems to be solved. We refer to this newly proposed enhancement as *solution checking.*

Lemma 9. *Let $1 \leqq \tilde{l} < l \leqq |E| + 1$ and let $q^{\tilde{l}}$ be an optimal solution for $\mathcal{P}(\tilde{l})$. If $q^{\tilde{l}}$ does not contain any of the elements $e_1, ..., e_{l-1}$, then it is optimal for $\mathcal{P}(l)$.*

Proof. We can find a solution of $\mathcal{P}(l)$ by solving a problem with the deterministic costs given in (3). For these costs we have

$$\begin{aligned}
&\tilde{l} \leqq l \Rightarrow \bar{\delta}_{\tilde{l}} \geqq \bar{\delta}_l \Rightarrow c^{\tilde{l}}(e_j) \leqq c^l(e_j) && \forall\ e_j : j < \tilde{l},\\
&j \leqq l \Rightarrow \delta_{e_j} \geqq \bar{\delta}_l \Rightarrow c^{\tilde{l}}(e_j) = \hat{c}_{e_j} \leqq \hat{c}_{e_j} + (\delta_{e_j} - \bar{\delta}_l) = c^l(e_j) && \forall\ e_j : \tilde{l} \leqq j < l,\\
&\tilde{l} \leqq l \Rightarrow c^{\tilde{l}}(e_j) = \hat{c}_{e_j} = c^l(e_j) && \forall\ e_j : j \geqq l.
\end{aligned}$$

If $q^{\tilde{l}}$ does not contain any element $e_j : j < l$, then

$$\sum_{e \in q^{\tilde{l}}} c^l(e) = \sum_{e \in q^{\tilde{l}}} c^{\tilde{l}}(e) \leqq \sum_{e \in q} c^{\tilde{l}}(e) \leqq \sum_{e \in q} c^l(e)\ \forall\ q \in Q,$$

Algorithm 2 Improved step 3 of Algorithm 1: Solve subproblems (with solution checking).

Input: $I = (E, Q, \hat{\boldsymbol{c}}, \boldsymbol{\delta}, \Gamma)$ with E ordered w.r.t. $\boldsymbol{\delta}$, $\bar{\boldsymbol{\delta}}$, an index set L of subproblems
Output: a set of solutions $\{q^l : l \in L\}$

```
1: $\tilde{l} := 0$
2: for all $l \in L$ in increasing order do
3:     if $\tilde{l} = 0$ or $q^{\tilde{l}}$ contains any element in $\{e_1, ..., e_{l-1}\}$ then
4:         Find an optimal solution $q^l$ for $(\mathcal{P}(l))$.
5:     else $q^l := q^{\tilde{l}}$
6:     end if
7:     $\tilde{l} := l$
8: end for
```

hence, $q^{\tilde{l}}$ is optimal for $\mathcal{P}(l)$. □

We can therefore replace Step 3 of the basic structure (Algorithm 1) with Algorithm 2.

Lemma 9 does not contain any theoretical complexity result since, in the worst case, still $\left\lceil \frac{|E|-\Gamma}{2} \right\rceil + 1$ subproblems are solved. Nevertheless, the results of our experiments in Section 4 show the practical use of this improvement.

3.1.2. The DSA for multi-objective problems

In this subsection, we extend the DSA to multi-objective problems. The idea presented in Subsection 3.1.1 is still valid. A set q has maximal cost in the i-th objective, if the cost of its Γ_i elements with the largest cost intervals $\delta_{e,i}$ take their maximal value. However, the sorting of the elements by interval lengths often results in a different order for each objective. An element that has the Γ_i-th longest interval in q for all $i = 1, ...k$ is not likely to exist. To ensure that the worst case vector of q equals the objective vector of a subproblem, we have to iterate through all elements for each objective independently and consider all possible combinations. The subproblems to be solved are hence constructed in the following way:
For $j = 1, ..., |E|, i = 1, ..., k$ let E^i_j be a set of the j elements with the largest intervals for the i-th objective with $E^i_1 \subset E^i_2 \subset ... \subset E^i_{|E|} = E$. I.e., $|E^i_j| = j$ and $\delta_{e,i} \geqq \delta_{e',i}\ \forall\ e \in E^i_j, e' \in E \backslash E^i_j$. We further define $\bar{\delta}^i_j := \min_{e \in E^i_j} \delta_{e,i}$ and $\bar{\delta}^i_{|E|+1} := 0\ \forall\ i$. For each $l = (l_1, ..., l_k) \in L := \{1, ..., |E|+1\} \times ... \times \{1, ..., |E|+1\}$ we define the problem

$$(\mathcal{MP}(\boldsymbol{l}))\ \min_{q \in Q} \boldsymbol{g}^{\boldsymbol{l}}(q) := \begin{pmatrix} \sum\limits_{e \in q} \hat{c}_{e,1} + \Gamma_1 \cdot \bar{\delta}^1_{l_1} + \sum\limits_{e \in q \cap E^1_{l_1}} (\delta_{e,1} - \bar{\delta}^1_{l_1}) \\ \vdots \\ \sum\limits_{e \in q} \hat{c}_{e,k} + \Gamma_k \cdot \bar{\delta}^k_{l_k} + \sum\limits_{e \in q \cap E^k_{l_k}} (\delta_{e,k} - \bar{\delta}^k_{l_k}) \end{pmatrix}.$$

We are now looking for a complete set of solutions for each of the subproblems. Such a solution set can be found by solving a deterministic multi-objective problem of the same kind as the original problem. We denote the solution set that we obtain for $\mathcal{MP}(\boldsymbol{l})$ by $OPT^{\boldsymbol{l}}$.
Algorithm 3 preserves the basic structure of DSA: First, the elements are sorted w.r.t. $\delta_{e,i}$ for each $i = 1, ..., k$. Instead of changing the indices, we store the set E^i_j of the first j elements for all $j = 1, ..., |E|$, because the order of the elements depends on the objective. Then the

Algorithm 3 DSA for general multi-objective instances

```
Input: an instance $I = (E, Q, \hat{C}, \Delta, \mathbf{\Gamma})$ of (MOUCO)
Output: a complete set of robust efficient solutions for $I$
1: For $i := 1, ..., k$: Sort $E$ w.r.t. $\delta_{e,i}$ descending and save the first $j$ elements in $E^i_j$ for $j = 1, ..., |E|$. Set $E^i_{|E|+1} := E$. Set $\bar{\delta}^i_j := \min_{e \in E^i_j} \delta_{e,i}\ \forall\ j = 1, ..., |E|$ and $\bar{\delta}^i_{|E|+1} := 0$.
2: Determine $L = L_1 \times L_2 \times ... \times L_k$: $L_i := \{1, ..., |E| + 1\}\ \forall\ i = 1, ..., k$.
3: For all $\boldsymbol{l} \in L$ find a complete set of efficient solutions $OPT^{\boldsymbol{l}}$ for $(\mathcal{MP}(\boldsymbol{l}))$.
4: Compare the objective vectors $\boldsymbol{z}^{\mathbf{R}}(q)$ of all solutions in $\cup_{\boldsymbol{l} \in L} OPT^{\boldsymbol{l}}$. The solutions with non-dominated objective vectors form a complete set of robust efficient solutions.
```

set L is determined, which contains vectors instead of scalar values. For each element in L the subproblem defined above is solved and their solutions are compared to obtain the robust efficient solutions.

Theorem 10. *Algorithm 3 finds a complete set of robust efficient solutions for (MOUCO).*

Proof. First, we show that $\boldsymbol{g}^{\boldsymbol{l}}$ never underestimates $\boldsymbol{z}^{\mathbf{R}}$ for any objective. Further, we prove that for each feasible solution q there is an $\boldsymbol{l} \in L$ with $\boldsymbol{g}^{\boldsymbol{l}}(q) = \boldsymbol{z}^{\mathbf{R}}(q)$. We conclude that Algorithm 3 finds a complete set of robust efficient solutions.
For each $q \in Q$, $\boldsymbol{l} \in L$ and $i \in \{1, ..., k\}$ we show $z^{\mathrm{R}}_i(q) \leqq g^{\boldsymbol{l}}_i(q)$. Let $\{e_{a_1}, \ldots, e_{a_h}\}$ be a set of h elements in q with the largest cost intervals $\delta_{e,i}$, where $h = \min\{|q|, \Gamma_i\}$. Then

$$\begin{aligned}
z^{\mathrm{R}}_i(q) = &\ \sum_{e \in q} \hat{c}_{e,i} + \sum_{j=1}^{h} (\delta_{e_{a_j},i} - \bar{\delta}^i_{l_i} + \bar{\delta}^i_{l_i}) \\
\leqq &\ \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}^i_{l_i} + \sum_{j=1}^{h} (\delta_{e_{a_j},i} - \bar{\delta}^i_{l_i}) && \text{since } h \leqq \Gamma_i \\
\leqq &\ \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}^i_{l_i} + \sum_{e \in q} \max\{0, \delta_{e,i} - \bar{\delta}^i_{l_i}\} && \text{since } \{e_{a_1}, \ldots, e_{a_h}\} \subseteq q \\
= &\ \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}^i_{l_i} + \sum_{e \in q \cap E^i_{l_i}} (\delta_{e,i} - \bar{\delta}^i_{l_i}) && \text{since } e \in E^i_{l_i} \Rightarrow \delta_{e,i} \geqq \bar{\delta}^i_{l_i}, e \notin E^i_{l_i} \Rightarrow \delta_{e,i} \leqq \bar{\delta}^i_{l_i} \\
= &\ g^{\boldsymbol{l}}_i(q).
\end{aligned}$$

We conclude $\boldsymbol{g}^{\boldsymbol{l}}(q) \leqq \boldsymbol{z}^{\mathbf{R}}(q)$ for all $q \in Q$ and $\boldsymbol{l} \in L$.
We show now that for every $q \in Q$ there is an $\bar{\boldsymbol{l}} \in L$ with $\boldsymbol{g}^{\bar{\boldsymbol{l}}}(q) = \boldsymbol{z}^{\mathbf{R}}(q)$. Given $q \in Q$ we construct $\bar{\boldsymbol{l}}$ as follows: For all $i \in \{1, ..., k\}$ with $\Gamma_i > |q|$, we set $\bar{l}_i := |E| + 1$, since

$$z^{\mathrm{R}}_i(q) = \sum_{e \in q} \hat{c}_{e,i} + \sum_{e \in q} \delta_{e,i} = \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot 0 + \sum_{e \in q} (\delta_{e,i} - 0).$$

For all $i \in \{1, ..., k\}$ with $\Gamma_i \leqq |q|$ we choose $\bar{l}_i$ such that $q \cap E^i_{\bar{l}_i}$ contains exactly Γ_i elements. These Γ_i elements have the largest cost intervals $\delta_{e,i}$ among all elements in q, i.e., the worst case cost for q is

$$\begin{aligned} z_i^{\mathrm{R}}(q) = & \sum_{e\in q} \hat{c}_{e,i} + \sum_{e\in q\cap E^i_{\bar{l}_i}} \delta_{e,i} \\ = & \sum_{e\in q} \hat{c}_{e,i} + \sum_{e\in q\cap E^i_{\bar{l}_i}} \bar{\delta}^i_{\bar{l}_i} + \sum_{e\in q\cap E^i_{\bar{l}_i}} (\delta_{e,i} - \bar{\delta}^i_{\bar{l}_i}) \\ = & \sum_{e\in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}^i_{\bar{l}_i} + \sum_{e\in q\cap E^i_{\bar{l}_i}} (\delta_{e,i} - \bar{\delta}^i_{\bar{l}_i}) \qquad \text{since } |q \cap E^i_{\bar{l}_i}| = \Gamma_i. \end{aligned}$$

We conclude $\boldsymbol{z}^{\mathbf{R}}(q) = \boldsymbol{g}^{\bar{\boldsymbol{l}}}(q)$. If q is robust efficient, then there is no $q' \in Q$ with $\boldsymbol{z}^{\mathbf{R}}(q') \leq \boldsymbol{z}^{\mathbf{R}}(q)$. It follows that

$$\nexists q' \in Q : \boldsymbol{z}^{\mathbf{R}}(q') \leqq \boldsymbol{z}^{\mathbf{R}}(q) \overset{\boldsymbol{z}^{\mathbf{R}}(q') \leqq \boldsymbol{g}^{\bar{\boldsymbol{l}}}(q')}{\Rightarrow} \nexists q' \in Q : \boldsymbol{g}^{\bar{\boldsymbol{l}}}(q') \leqq \boldsymbol{z}^{\mathbf{R}}(q) = \boldsymbol{g}^{\bar{\boldsymbol{l}}}(q).$$

Therefore, q or an equivalent solution is found at least once in the algorithm. It follows that in Step 4 the objective vector of each found solution is compared to all non-dominated objective vectors, thus only robust efficient solutions remain. It follows that the output is a complete set of robust efficient solutions. □

Example 11. *Consider the instance in Example 7 (Figure 1). In Step 1 of Algorithm 3 we obtain*

$$\bar{\delta}^1 = (5,4,3,3,3,2,1,1,1)^T, \bar{\delta}^2 = (5,5,5,5,5,5,1,1,1)^T$$

and for example

$$\begin{array}{llll} E^1_1 = \{(v_2,v_5)\} & E^1_2 = E^1_1 \cup \{(v_2,v_4)\} & E^1_3 = E^1_2 \cup \{(v_1,v_3)\} & E^1_4 = E^1_2 \cup \{(v_3,v_4)\} \\ E^1_5 = E^1_4 \cup \{(v_5,v_6)\} & E^1_6 = E^1_5 \cup \{(v_3,v_5)\} & E^1_7 = E^1_6 \cup \{(v_1,v_2)\} & E^1_8 = E^1_7 \cup \{(v_2,v_3)\} \\ E^1_9 = E^1_8 \cup \{(v_4,v_6)\} & & & \\ E^2_1 = \{(v_2,v_5)\} & E^2_2 = E^2_1 \cup \{(v_2,v_4)\} & E^2_3 = E^2_2 \cup \{(v_1,v_3)\} & E^2_4 = E^2_2 \cup \{(v_3,v_4)\} \\ E^2_5 = E^2_4 \cup \{(v_5,v_6)\} & E^2_6 = E^2_5 \cup \{(v_3,v_5)\} & E^2_7 = E^2_6 \cup \{(v_1,v_2)\} & E^2_8 = E^2_7 \cup \{(v_2,v_3)\} \\ E^2_9 = E^2_8 \cup \{(v_4,v_6)\}. & & & \end{array}$$

Step 2 sets $L := \{1,...,9\} \times \{1,...,9\}$ and in Step 3 ($\mathcal{MP}(\boldsymbol{l})$) is solved for all $\boldsymbol{l} \in L$.
As an example, we consider $\boldsymbol{l} = (7,8)$. Recall, that the path $q_1 := ((v_1,v_2),(v_2,v_4),(v_4,v_6))$ is robust efficient. Since $|E^1_7 \cap q_1| = 2$ and $|E^2_8 \cap q_1| = 2$, we know from the proof of Theorem 10 that $\boldsymbol{g}^{(\mathbf{7},\mathbf{8})}(q_1) = \boldsymbol{z}^{\boldsymbol{R}}(q_1)$ and that q_1 is an efficient solution for

$$(\mathcal{MP}(7,8)) \ \min_{q\in Q} \boldsymbol{g}^{(\mathbf{7},\mathbf{8})}(q) := \begin{pmatrix} \sum_{e\in q} \hat{c}_{e,1} + \Gamma_1 \cdot \bar{\delta}^1_7 + \sum_{e\in q\cap E^1_7} (\delta_{e,1} - \bar{\delta}^1_7) \\ \sum_{e\in q} \hat{c}_{e,2} + \Gamma_2 \cdot \bar{\delta}^2_8 + \sum_{e\in q\cap E^2_8} (\delta_{e,2} - \bar{\delta}^2_8) \end{pmatrix}.$$

A complete set of efficient solutions $OPT^{(7,8)}$ for ($\mathcal{MP}(7,8)$) can be obtained by solving the instance of the deterministic multi-objective shortest path problem shown in Figure 2. The edge costs are

$$c_1^{(7,8)}(e) := \begin{cases} \hat{c}_{e,1} + \delta_{e,1} - \bar{\delta}^1_7 & \text{if } e \in E^1_7 \\ \hat{c}_{e,1} & \text{else} \end{cases} \qquad c_2^{(7,8)}(e) := \begin{cases} \hat{c}_{e,2} + \delta_{e,2} - \bar{\delta}^2_8 & \text{if } e \in E^2_8 \\ \hat{c}_{e,2} & \text{else.} \end{cases}$$

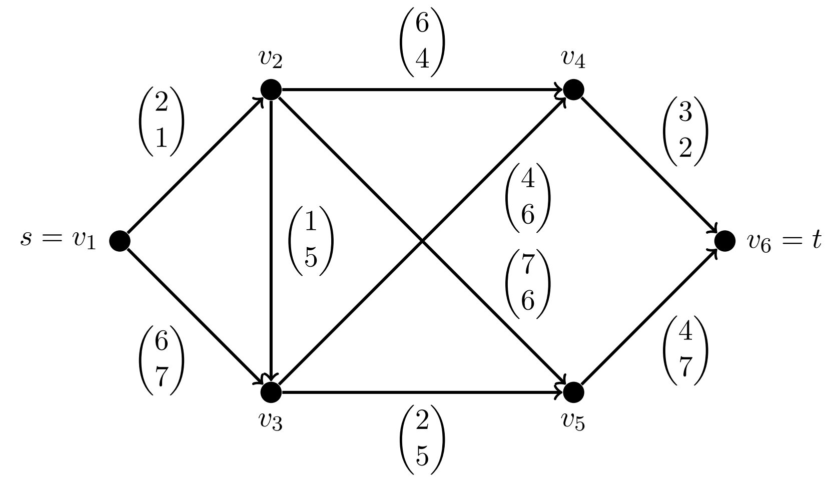


Figure 2: $OPT^{(7,8)}$ in Example 11 is obtained by solving this instance of the multi-objective shortest path problem.

The path q_1 is indeed efficient for this instance with $\boldsymbol{c}^{\mathbf{(7,8)}}(q_1) = (11, 7)^T$. It follows

$$\boldsymbol{g}^{\mathbf{(7,8)}}(q_1) = \begin{pmatrix} \Gamma_1 \cdot \bar{\delta}^1_7 + 11 \\ \Gamma_2 \cdot \bar{\delta}^2_8 + 7 \end{pmatrix} = \begin{pmatrix} 13 \\ 9 \end{pmatrix} = \boldsymbol{z}^{\boldsymbol{R}}(q_1).$$

The path $q_3 := \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_6)\}$ is efficient for this instance as well, hence $q_1, q_3 \in OPT^{(7,8)}$.
In Step 4 of Algorithm 3, all obtained solutions are compared to each other. The path q_3 is not robust efficient, because $\boldsymbol{z}^{\boldsymbol{R}}(q_2) = (11, 16)^T \leq (12, 16)^T = \boldsymbol{z}^{\boldsymbol{R}}(q_3)$. Since $q_2 \in OPT^{(4,4)}$, $\boldsymbol{z}^{\boldsymbol{R}}(q_2)$ and $\boldsymbol{z}^{\boldsymbol{R}}(q_3)$ are compared to each other in Step 4 and the returned solution set does not contain q_3. However, it contains q_1, because q_1 is robust efficient and hence there does not exist any path q' with $\boldsymbol{z}^{\boldsymbol{R}}(q') \leq \boldsymbol{z}^{\boldsymbol{R}}(q_1)$.

As for the single-objective version, we can reduce the number of subproblems to be solved. The results of Lemma 8 are still valid for each objective independently. Therefore we can replace the L_i as described in the following lemma.

Lemma 12. *The number of subproblems to be solved by Algorithm 3 can be reduced to $\prod_{i=1}^k \left(\left\lceil \frac{|E|-\Gamma_i}{2} \right\rceil + 1 \right)$ in the same ways as in the single-objective case (Lemma 8):*

1. *Let $i \in \{1, ..., k\}$ be given. If there are several elements with the same interval length $\delta_{e,i}$, i.e., there exist pairwise different indices $j_1, ..., j_h \in \{1, ..., |E|\}$ with $\bar{\delta}^i_{j_1} = ... = \bar{\delta}^i_{j_h}$, then it is sufficient that l_i takes one of the values in $\{j_1, ..., j_h\}$.*
2. *For all $i \in \{1, ..., k\}$ it is sufficient, that l_i takes every second value in $\{1, .., |E|\}$ and the value $|E| + 1$.*
3. *It is sufficient that l_i takes values that are greater or equal to Γ_i.*

Proof.

1. Let $\hat{l}_1, \hat{l}_2, ..., \hat{l}_{i-1}, \hat{l}_{i+1}, ..., \hat{l}_k \in \{1, ..., |E| + 1\}$ be fixed values. We define the vector $\hat{\boldsymbol{l}}^{\boldsymbol{x}} := (\hat{l}_1, \hat{l}_2, ..., \hat{l}_{i-1}, x, \hat{l}_{i+1}, ..., \hat{l}_k)$. From $\bar{\delta}^i_{j_1} = ... = \bar{\delta}^i_{j_h}$ it follows directly

$$\sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}^i_{j_1} + \sum_{e \in q \cap E^i_{j_1}} (\delta_{e,i} - \bar{\delta}^i_{j_1}) = \ldots = \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}^i_{j_h} + \sum_{e \in q \cap E^i_{j_h}} (\delta_{e,i} - \bar{\delta}^i_{j_h})$$

and therefore $\mathcal{MP}(\hat{\boldsymbol{l}}^{\boldsymbol{j_1}}) = ... = \mathcal{MP}(\hat{\boldsymbol{l}}^{\boldsymbol{j_h}})$.

2. Let $q \in Q$ be a feasible solution. We have shown in the proof of Theorem 10 that there exists an $\bar{\boldsymbol{l}} \in L$ with $\boldsymbol{z}^{\mathbf{R}}(q) = \boldsymbol{g}^{\bar{\boldsymbol{l}}}(q)$ and either $\bar{l}_i = |E| + 1$ or $\Gamma_i = |q \cap E^i_{\bar{l}_i}|$. In the second case, since $q \cap E^i_{\bar{l}_i}$ contains the Γ_i elements in q with the largest cost intervals $\delta_{e,i}$, we have

$$\begin{aligned} z^{\mathrm{R}}_i(q) = & \sum_{e\in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}^i_{\bar{l}_i} + \sum_{e \in q \cap E^i_{\bar{l}_i}} (\delta_{e,i} - \bar{\delta}^i_{\bar{l}_i}) \\ = & \sum_{e\in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}^i_{\bar{l}_i} + \sum_{e \in q \cap E^i_{\bar{l}_i}} (\delta_{e,i} - \bar{\delta}^i_{\bar{l}_i}) + \Gamma_i \cdot (\bar{\delta}^i_{(\bar{l}_i+1)} - \bar{\delta}^i_{\bar{l}_i}) + \Gamma_i \cdot (\bar{\delta}^i_{\bar{l}_i} - \bar{\delta}^i_{(\bar{l}_i+1)}) \\ = & \sum_{e\in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}^i_{(\bar{l}_i+1)} + \sum_{e \in q \cap E^i_{\bar{l}_i}} (\delta_{e,i} - \bar{\delta}^i_{(\bar{l}_i+1)}), \qquad \text{because } |q \cap E^i_{\bar{l}_i}| = \Gamma_i \\ = & \sum_{e\in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}^i_{(\bar{l}_i+1)} + \sum_{e \in q \cap E^i_{(\bar{l}_i+1)}} (\delta_{e,i} - \bar{\delta}^i_{(\bar{l}_i+1)}), \end{aligned}$$

because $\delta_{e,i} = \bar{\delta}^i_{(\bar{l}_i+1)}$ for $e \in E^i_{(\bar{l}_i+1)} \setminus E^i_{\bar{l}_i}$. Therefore, if $\Gamma_i \leqq |q|$, it is sufficient that l_i either takes the value $\bar{l}_i$ or $\bar{l}_i + 1$. If $\Gamma_i \geqq |q|$, it is sufficient that l_i takes the value $|E| + 1$.

3. In the proof of Theorem 10 we have show that for every $q \in Q$ there is an $\bar{\boldsymbol{l}} \in L$ with $\boldsymbol{z}^{\mathbf{R}}(q) = \boldsymbol{g}^{\bar{\boldsymbol{l}}}(q)$ and either $\bar{l}_i = |E| + 1$ or $\Gamma_i = |q \cap E^i_{\bar{l}_i}| \leqq |E^i_{\bar{l}_i}| = \bar{l}_i$.

From statement 3 we know that l_i takes at most $|E| + 1 - (\Gamma_i - 1)$ different values. From statement 2 it follows that of these values the last one and every second of the other ones are sufficient. This leads to at most

$$\left\lfloor \frac{|E| + 1 - (\Gamma_i - 1) - 1}{2} \right\rfloor + 1 = \left\lfloor \frac{|E| - \Gamma_i + 1}{2} \right\rfloor + 1 = \left\lceil \frac{|E| - \Gamma_i}{2} \right\rceil + 1$$

different values of l_i. Therefore, it is sufficient to solve $\prod_{i=1}^k \left(\left\lceil \frac{|E|-\Gamma_i}{2} \right\rceil + 1 \right)$ subproblems. $\square$

Here again, we can use solution checking, i.e., skip some additional subproblems, depending on the solutions found so far. However, we now have to ensure that $\tilde{\boldsymbol{l}} \leq \boldsymbol{l}$ and that none of the solutions in $OPT^{\tilde{\boldsymbol{l}}}$ contains any of the elements, whose costs have been increased.

Lemma 13. *Let $\boldsymbol{l}, \tilde{\boldsymbol{l}} \in \mathbb{Z}^k$ be given with $\tilde{\boldsymbol{l}} \leq \boldsymbol{l}$ and let J be the set of indices i with $\tilde{l}_i < l_i$. Let $OPT^{\tilde{\boldsymbol{l}}}$ be a complete set of efficient solutions for $\mathcal{MP}(\tilde{\boldsymbol{l}})$. If none of the sets in $OPT^{\tilde{\boldsymbol{l}}}$ contains an element in $\cup_{i \in J} E^i_{l_i}$, then $OPT^{\tilde{\boldsymbol{l}}}$ is a complete set of efficient solutions for $\mathcal{MP}(\boldsymbol{l})$.*

Proof. Since $\Gamma_i \cdot \bar{\delta}^{l_i}_i$ are solution independent constants, the minimization problem to be solved is a deterministic multi-objective problem with costs $\boldsymbol{c}^{\boldsymbol{l}}(e) = (c^{\boldsymbol{l}}_1(e), ..., c^{\boldsymbol{l}}_k(e))$:

$$c^{\boldsymbol{l}}_i(e) := \begin{cases} \hat{c}_{e,i} + (\delta_{e,i} - \bar{\delta}^i_{l_i}) & \text{for } e \in E^i_{l_i} \\ \hat{c}_{e,i} & \text{else.} \end{cases}$$

Algorithm 4 Improved Step 3 of Algorithm 3: Solve subproblems (with solution checking).

```
Input: an instance $I = (E, Q, \hat{C}, \Delta, \mathbf{\Gamma})$, $\bar{\delta}_i^j$ and $E_j^i$ $\forall\ i,j \in \{1,...,k\}$, an index set $L$ of
    subproblems
Output: solution sets $(OPT^{\boldsymbol{l}}, \boldsymbol{l} \in L)$
 1: $\tilde{\boldsymbol{l}}^{\mathbf{1}} := (0,...,0)$
 2: $h := 1$
 3: for all $l_1 \in L_1$ in increasing order do
 4:     for all $l_2 \in L_2$ in increasing order do
 5:         ...
 6:         for all $l_k \in L_k$ in increasing order do
 7:             $\boldsymbol{l} := (l_1, ..., l_k)$
 8:             if $\tilde{\boldsymbol{l}}^{\boldsymbol{h}} = (0,...,0)$ or any of the sets in $OPT^{\tilde{\boldsymbol{l}}^{\boldsymbol{h}}}$ contains any element in $E_{l_h}^h$ then
 9:                 Find a complete set of efficient solutions $OPT^{\boldsymbol{l}}$ for $(\mathcal{MP}(\boldsymbol{l}))$.
10:             else $OPT^{\boldsymbol{l}} := OPT^{\tilde{\boldsymbol{l}}^{\boldsymbol{h}}}$
11:             end if
12:             for $i = h, ..., k$ do
13:                 $\tilde{\boldsymbol{l}}^{\boldsymbol{i}} := \boldsymbol{l}$
14:             end for
15:             $h := k$
16:         end for
17:         ...
18:         $h := 2$
19:     end for
20:     $h := 1$
21: end for
```

Since $\tilde{l}_i \leqq l_i \Rightarrow \bar{\delta}_{\tilde{l}}^i \geqq \bar{\delta}_l^i$, it follows

$$\begin{aligned}
c_i^{\tilde{\boldsymbol{l}}}(e) &= c_i^{\boldsymbol{l}}(e)\ \forall\ i \text{ with } l_i = \tilde{l}_i,\ \forall\ e \in E \\
c_i^{\tilde{\boldsymbol{l}}}(e) &= c_i^{\boldsymbol{l}}(e)\ \forall\ i \text{ with } \tilde{l}_i < l_i,\ \forall\ e \in E \setminus E_{l_i}^i \\
c_i^{\tilde{\boldsymbol{l}}}(e) &\leqq c_i^{\boldsymbol{l}}(e)\ \forall\ i,\ \forall\ e \in E.
\end{aligned}$$

Hence, if none of the sets in $OPT^{\tilde{\boldsymbol{l}}}$ contains any element in $\cup_{i \in J} E_{l_i}^i$, we have $c_i^{\tilde{\boldsymbol{l}}}(e) = c_i^{\boldsymbol{l}}(e)$ for all elements that are contained in any set in $OPT^{\tilde{\boldsymbol{l}}}$, and $c_i^{\tilde{\boldsymbol{l}}}(e) \leqq c_i^{\boldsymbol{l}}(e)$ for all elements in E. It follows, that every $q \in OPT^{\tilde{\boldsymbol{l}}}$ is also efficient w.r.t $\boldsymbol{c}^{\boldsymbol{l}}$. Furthermore, for every $q' \notin OPT^{\tilde{\boldsymbol{l}}}$ exists a $q \in OPT^{\tilde{\boldsymbol{l}}}$ with

$$\sum_{e \in q} \boldsymbol{c}^{\boldsymbol{l}}(e) = \sum_{e \in q} \boldsymbol{c}^{\tilde{\boldsymbol{l}}}(e) \leqq \sum_{e \in q'} \boldsymbol{c}^{\tilde{\boldsymbol{l}}}(e) \leqq \sum_{e \in q'} \boldsymbol{c}^{\boldsymbol{l}}(e),$$

so q' is either dominated w.r.t. $\boldsymbol{c}^{\boldsymbol{l}}$ or has an equivalent solution in $OPT^{\tilde{\boldsymbol{l}}}$. Therefore, $OPT^{\tilde{\boldsymbol{l}}}$ is a complete set of efficient solutions for $\mathcal{MP}(\boldsymbol{l})$. □

A fast way to use this result is to replace Step 3 of Algorithm 3 with Algorithm 4.

We loop through all $\boldsymbol{l} \in L$. In Lines 8 to 11, $OPT^{\boldsymbol{l}}$ is found for the current $\boldsymbol{l}$: Either $(\mathcal{MP}(\boldsymbol{l}))$ is solved, or $OPT^{\boldsymbol{l}}$ is set to the solution set of an already solved subproblem. For this purpose, we store one vector $\tilde{\boldsymbol{l}}^{\boldsymbol{h}}$ for each $h = 1, ..., k$, which is updated in Line 13 whenever the value l_h has changed, i.e. whenever l_i was increased for some $i \leqq h$ in the respective for-loop. When l_h is increased in the for-loop, during the next execution of Line 8 we have

$$\tilde{l}_i^h = \begin{cases} l_i & \text{for } i < h, \text{ because } \tilde{\boldsymbol{l}}^{\boldsymbol{h}} \text{ was updated after the previous change of } l_i, \\ l_i - 1 & \text{for } i = h, \text{ because } l_h \text{ was increased, but } \tilde{\boldsymbol{l}}^{\boldsymbol{h}} \text{ is not updated yet,} \\ 1 = l_i & \text{for } i > h, \text{ as, due to the nested for-loops, } l_i \text{ is set to 1 whenever } l_h \text{ changes.} \end{cases}$$

Hence, if no set in $OPT^{\tilde{\boldsymbol{l}}^{\boldsymbol{h}}}$ contains any element in $E_{l_h}^h$ the conditions of Lemma 13 are satisfied for $\tilde{\boldsymbol{l}} := \tilde{\boldsymbol{l}}^{\boldsymbol{h}}$.

Corollary 14. *Algorithm 3, with Algorithm 4 replacing Step 3 and the construction of L (Step 2) adjusted according to Lemma 12, finds a complete set of robust efficient solutions for (MOUCO). During its execution at most* $\prod_{i=1}^k \left(\left\lceil \frac{|E|-\Gamma_i}{2} \right\rceil + 1 \right)$ *deterministic subproblems have to be solved.*

For problems with the following property, the number of subproblems to be solved can be reduced significantly.

Definition 15. *An instance* $(E, Q, \hat{C}, \Delta, \boldsymbol{\Gamma})$ *has* partial objective-independent element order *if there exists a subset* $J := \{i_1, ..., i_r\} \subseteq \{1, ..., k\}$ *with*

- $\Gamma_{i_1} = \Gamma_{i_2} = ... = \Gamma_{i_r}$ *and*
- *there exists an order of the elements in E, such that*

$$\delta_{e_1,i} \geqq ... \geqq \delta_{e_{|E|},i} \ \forall \ i \in J.$$

If $J = \{1, ..., k\}$*, the instance has* objective-independent element order.

Example 16. *Consider an instance with* $E = \{e_1, e_2, e_3\}$ *and*

$$\boldsymbol{\delta_{e_1}} = (1, 1, 1)^T, \boldsymbol{\delta_{e_2}} = (3, 2, 1)^T, \boldsymbol{\delta_{e_3}} = (2, 2, 1)^T.$$

Then $\delta_{e_1,i} \leq \delta_{e_3,i} \leq \delta_{e_2,i} \ \forall \ i = 1, ..., 3$*, hence the instance has objective-independent element order. With*

$$\boldsymbol{\delta_{e_1}} = (1, 2, 3)^T, \boldsymbol{\delta_{e_2}} = (3, 2, 1)^T, \boldsymbol{\delta_{e_3}} = (2, 2, 2)^T$$

the instance does not have objective-independent element order, because $\delta_{e_1,1} < \delta_{e_3,1}$ *and* $\delta_{e_3,3} < \delta_{e_1,3}$*. However, it has partial objective-independent element order, because, e.g.,* $\delta_{e_2,i} \leqq \delta_{e_3,i} \leqq \delta_{e_1,i}$ *for* $i = 2, 3$*.*

Lemma 17. *Let an instance* $(E, Q, \hat{C}, \Delta, \boldsymbol{\Gamma})$ *with partial objective-independent element order be given and let J be the set of indices defined in Definition 15. Then the nested for-loops changing* $l_{i_1}, ..., l_{i_r}$ *in Algorithm 4 can be replaced by a single for-loop. The number of solved deterministic subproblems in Algorithm 3 with Algorithm 4 (with replaced for-loops) as Step 3 and L adjusted according to Lemma 12 is then less than or equal to*

$$\begin{cases} \left(\left\lceil \dfrac{|E| - \Gamma_{i_1}}{2} \right\rceil + 1 \right) & \textit{if } J = \{1, ..., k\} \\ \left(\left\lceil \dfrac{|E| - \Gamma_{i_1}}{2} \right\rceil + 1 \right) \cdot \prod\limits_{i \in \{1,...,k\} \setminus \{i_1,...,i_r\}} \left(\left\lceil \dfrac{|E| - \Gamma_i}{2} \right\rceil + 1 \right) & \textit{otherwise.} \end{cases}$$

Proof. In the proof of Theorem 10 we have shown that for each $q \in Q$ there exists an $\boldsymbol{l} \in L$ with $\boldsymbol{z}^{\mathbf{R}}(q) = \boldsymbol{g}^{\boldsymbol{l}}(q)$. We show that there always is such an $\boldsymbol{l}$ with $l_{i_1} = ... = l_{i_r}$.
Since there exists an order of the elements in E such that $\delta_{e_1,i} \geqq ... \geqq \delta_{e_{|E|},i} \ \forall \ i \in J$, we can choose the sets E^i_j such that $E^{i_1}_j = ... = E^{i_r}_j \ \forall \ j = 1, ..., |E|$. In the proof of Theorem 10 we choose $\bar{l}_i$ such that $E^i_{\bar{l}_i} \cap q$ has exactly Γ_i elements. With $\Gamma_{i_1} = ... = \Gamma_{i_r}$ it follows $\bar{l}_{i_1} = ... = \bar{l}_{i_r}$. Hence, we have $\boldsymbol{z}^{\mathbf{R}}(q) = \boldsymbol{g}^{\bar{\boldsymbol{l}}}(q)$ and $\bar{l}_{i_1} = ... = \bar{l}_{i_r}$.
It follows that the nested for-loops changing $l_{i_1}, ..., l_{i_r}$ can be replaced by a single for-loop, which leads directly to the stated number of subproblems. □

3.2. Bottleneck approach

In the algorithms presented in the previous section, the number of subproblems that have to be solved increases with decreasing values of Γ_i. In this section we present a method whose complexity decreases with decreasing values of Γ_i. Its idea is to transfer (MOUCO) with k objectives into a deterministic combinatorial optimization problem of the same kind with $\sum_{i=1}^k (\Gamma_i + 1)$ objective functions, some of which are bottleneck functions instead of sum functions. The concept is particularly useful if an efficient algorithm for solving the deterministic multi-objective problem with sum and bottleneck functions is available. As an example we present such an algorithm for the shortest path problem in Section 3.2.2.

3.2.1. Bottleneck approach for cardinality-constrained uncertain combinatorial optimization problems

We first explain the approach for the single-objective uncertain problem $(\min_{q \in Q} z(q), c \in \mathcal{U}^{cc})$ with $\mathcal{U}^{cc}$ as given in Equation (1). The robust counterpart (MORCO) then reduces to

$$(RCO) \qquad \min_{q \in Q} \left(z^{\mathrm{R}}(q) = \max_{c \in \mathcal{U}^{cc}} \sum_{e \in q} c(e) \right).$$

Definition 18. *For a subset $q \subseteq E$ and given interval lengths δ_e for all $e \in E$, we sort the elements in q by decreasing interval lengths and define $j\text{-}\max_{e \in q} \delta_e$ as the interval length of the j-th element according to this sorting.*

Theorem 19. *Every optimal solution for (RCO) is an efficient solution for the deterministic multi-objective problem*

$$(DCO) \qquad \min_{q \in Q} \left(\boldsymbol{z}^{\boldsymbol{D}}(q) := \begin{pmatrix} \sum_{e \in q} \hat{c}_e \\ \max_{e \in q} \delta_e \\ 2\text{-}\max_{e \in q} \delta_e \\ \vdots \\ \Gamma\text{-}\max_{e \in q} \delta_e \end{pmatrix} \right).$$

Proof. Recall that any feasible set $q \in Q$ has maximal cost if the cost of its Γ elements with the largest cost intervals take their maximal values. Let q be an optimal solution for (RCO). Assume that q is not efficient for (DCO). Then there exists a solution $q' \in Q$ that dominates

q and it follows

$$\sum_{e\in q'} \hat{c}_e \leqq \sum_{e\in q} \hat{c}_e \text{ and } j\text{-}\max_{e\in q'} \delta_e \leqq j\text{-}\max_{e\in q} \delta_e \ \forall\ j = 1, ..., \Gamma, \text{ with at least one inequality}$$

$$\Rightarrow z^{\mathrm{R}}(q') = \sum_{e\in q'} \hat{c}_e + \sum_{j=1}^{\Gamma} j\text{-}\max_{e\in q'} \delta_e < \sum_{e\in q} \hat{c}_e + \sum_{j=1}^{\Gamma} j\text{-}\max_{e\in q} \delta_e = z^{\mathrm{R}}(q).$$

This contradicts q being optimal for (RCO). □

The reverse of Theorem 19 does not hold: There exist efficient solutions for (DCO), which are not optimal for (RCO), as the following example shows.

Example 20. *Let G be a graph that consists of two disjoint paths q, q' from s to t with three edges each. Let the cost interval of all edges in q be $[1, 1]$ and of all edges in q' be $[0, 1]$ and let $\Gamma = 2$. Then both paths are efficient solutions for (DCO), because*

$$\boldsymbol{z^D}(q) = (3, 0, 0) \nleq (0, 1, 1) = \boldsymbol{z^D}(q') \text{ and } \boldsymbol{z^D}(q') = (0, 1, 1) \nleq (3, 0, 0) = \boldsymbol{z^D}(q).$$

But only q' is robust efficient, because

$$z^R(q') = 2 < 3 = z^R(q).$$

Lemma 21. *A complete set of efficient solutions for (DCO) contains at least one optimal solution for (RCO).*

Proof. Let $Q' \subseteq Q$ be a complete set of efficient solutions for (DCO). Assume, that (RCO) has an optimal solution q that is not contained in Q'. According to Theorem 19, q is an efficient solution for (DCO), so Q' contains a solution q' with

$$\begin{pmatrix} \sum_{e\in q} \hat{c}_e \\ \max_{e\in q} \delta_e \\ 2\text{-}\max_{e\in q} \delta_e \\ \vdots \\ \Gamma\text{-}\max_{e\in q} \delta_e \end{pmatrix} = \begin{pmatrix} \sum_{e\in q'} \hat{c}_e \\ \max_{e\in q'} \delta_e \\ 2\text{-}\max_{e\in q'} \delta_e \\ \vdots \\ \Gamma\text{-}\max_{e\in q'} \delta_e \end{pmatrix} \Rightarrow z^{\mathrm{R}}(q) = \sum_{e\in q} \hat{c}_e + \sum_{j=1}^{\Gamma} j\text{-}\max_{e\in q} \delta_e = z^{\mathrm{R}}(q')$$

and q' is optimal for (RCO). □

Now, we transfer this approach to the multi-objective case. For a problem with k objectives, we construct a deterministic problem with $m := \sum_{i=1}^{k}(\Gamma_i + 1)$ objectives.

Theorem 22. *Every efficient solution for the multi-objective robust counterpart (MORCO) is an efficient solution for the deterministic multi-objective problem*

$$(MODCO) \qquad \min_{q\in Q} \left(\boldsymbol{z^D}(q) := \begin{pmatrix} \sum_{e\in q} \hat{c}_{e,1} \\ \max_{e\in q} \delta_{e,1} \\ 2\text{-}\max_{e\in q} \delta_{e,1} \\ \vdots \\ \Gamma_1\text{-}\max_{e\in q} \delta_{e,1} \\ \sum_{e\in q} \hat{c}_{e,2} \\ \max_{e\in q} \delta_{e,2} \\ \vdots \\ \Gamma_k\text{-}\max_{e\in q} \delta_{e,k} \end{pmatrix} \right).$$

A complete set of solutions for (MODCO) contains a complete set of solutions for (MORCO).

Proof. Let q be an efficient solution for (MORCO). Assume that q is not efficient for (MODCO). Analogously to the proof of Theorem 19, there is a solution $q' \in Q$ dominating q and it follows that $z_i^{\mathrm{R}}(q') < z_i^{\mathrm{R}}(q)$ for at least one $i \in \{1, ..., k\}$, which contradicts q being efficient for (MORCO).
Assume now, that $q \notin Q'$ with Q' being a complete set of efficient solutions for (MODCO). Since q is efficient for (MODCO), there is a solution $q' \in Q'$ equivalent to q w.r.t. the objective function of (MODCO) and it follows $\boldsymbol{z}^{\mathbf{R}}(q) = \boldsymbol{z}^{\mathbf{R}}(q')$ analogously to the proof of Lemma 21. $\square$

With an algorithm to solve (MODCO) and a method to filter the obtained solutions we can now find a complete set of robust efficient solutions for the uncertain problem. In the case of a single-objective uncertain problem, Gorski et al. (2012) introduced an algorithm to solve (DCO).

3.2.2. Label setting algorithm (LSA) for (MOUSP)

In this section, we show how to apply the bottleneck approach to the cardinality-constrained uncertain shortest path problem. We propose an adjustment of standard multi-objective labeling algorithms (label setting or label correcting) to find a complete set of robust efficient solutions.
Let (MOUSP) be defined as in Section 2.1, i.e., E is the edge set of a graph and Q the set of simple paths from a given start node s to a given end node t. Additionally we assume non-negative edge costs ($\boldsymbol{c}(e) \geqq 0 \ \forall \ e \in E, \boldsymbol{c} \in \mathcal{U}^{mcc}$) and adjust a label setting algorithm as an example.
We first recall the definition of a label, which is used in common multi-objective labeling algorithms. A label $l = (\boldsymbol{y}, v', l')$ at a node v consists of

- a cost vector $\boldsymbol{y}$, here $\boldsymbol{y} = (y_1, ..., y_m)^T$,
- a predecessor node v', and
- a predecessor label l'.

Every label at a node $v \neq s$ with predecessor node v' *represents* a path q from s to v whose last edge is (v', v). That means that its cost equals the cost of q and its predecessor label l' represents the subpath of q from s to v'. We assume here, that no parallel edges exist, such that v and v' uniquely define an edge (v', v). If parallel edges have to be considered, the respective edge can be contained in the label as well. The labels are constructed iteratively from existing labels at the predecessor nodes and can at any time be either *temporary* or *permanent*.
Algorithm 5 is a label setting algorithm for solving (MODCO) for the shortest path problem. It is based on the label setting algorithm by Martins (1984) for multi-objective shortest path problems, but we make the following adjustments:

1. In Step 4 a label must be chosen whose cost is not dominated by the cost of any other temporary label. In the algorithm by Martins (1984) the lexicographically smallest label is chosen. Based on Iori et al. (2010), we choose the label with the smallest aggregate cost function $\sum_{j=1,...,m} y_j$ instead.
2. In multi-objective label setting algorithms with only sum functions (as considered by Martins (1984)) a new label $l = (\boldsymbol{y}, v', l')$ at v is created by adding the cost $\boldsymbol{y'}$ of

Algorithm 5 Label setting algorithm to solve (MODCO) for the shortest path problem

```
Input: an instance $I = (E, Q, \hat{C}, \Delta, \mathbf{\Gamma})$ of (MOUSP)
Output: permanent labels at $t$, representing a complete set of efficient solutions for instance $I$ of (MODCO)
1: Set $m := \sum_{i=1,...,k}(\Gamma_i + 1)$.
2: Create a temporary label $l_0$ with cost $(0, ..., 0)^T$ at node $s$.
3: while there exists at least one temporary label do
4:     Select a temporary label $l'$ (at any node $v'$) with minimal aggregate cost $\sum_{j=1,...,m} y'_j$ and make it permanent.
5:     for all outgoing edges $(v', v)$ of $v'$ do
6:         Create a new temporary label $l$ at $v$ by Algorithm 6.
7:         if the cost of $l$ is dominated by or equal to the cost of another label at $v$ then
8:             Delete $l$.
9:         else if $l$ dominates any temporary labels at $v$ then
10:            Delete these labels.
11:        end if
12:    end for
13: end while
```

the predecessor label l' to the edge cost. For min-max functions the (entry-wise) maximum of the edge cost and the predecessor label's cost is taken (see Gandibleux et al., 2006). To solve (MODCO) we need a new way to construct the labels: Let $n_i := 1 + \sum_{j=1,...,(i-1)}(\Gamma_j + 1)$ denote the index of the first objective of (MODCO) associated with the original objective z_i of (MORCO). For the sum objective functions, we add the nominal cost $\hat{c}_{e,i}$ of the edge $e := (v', v)$ to the corresponding predecessor cost entry y'_{n_i}. For the j- max objective functions, we compare for each objective z_i the interval length $\delta_{e,i}$ of e to each of the Γ_i longest interval lengths so far $y'_{n_i+1}, ..., y'_{n_i+\Gamma_i}$ and insert it at the right position (see Algorithm 6). We will use the following notation: $\boldsymbol{y} := \boldsymbol{y'} \oplus (\boldsymbol{\hat{c}_e}, \boldsymbol{\delta_e})$.

3. In the algorithm by Martins (1984) a newly created label is only deleted if it is dominated by a label at the same node. We delete the new label even if another label with equal cost exists at the same node, because we are only looking for a complete set of efficient solutions. This is also the reason why we do not need to consider *hidden* labels, which Gandibleux et al. (2006) introduced for problems with bottleneck functions. Since new labels with the same cost as existing labels are immediately deleted, Algorithm 5 works even without the assumption that no cycles of cost $(0, ..., 0)$ exist.

Example 23. *We show the first steps of Algorithm 5 with the instance given in Example 7 as input.*

1. *In Lines 1 and 2, m is set to* $(2 + 1) + (2 + 1) = 6$ *and a temporary label* l_0 *with cost* $(0, 0, 0, 0, 0, 0)^T$ *is created at node* v_1.
2. *The label* l_0 *is made permanent in Line 4 and new temporary labels are created at the nodes* v_2, v_3*:*

$$l_2^1 \textit{ at } v_2 \textit{ with cost } (2, 1, 0, 1, 1, 0)^T \textit{ representing } \{(v_1, v_2)\}$$
$$l_3^1 \textit{ at } v_3 \textit{ with cost } (4, 3, 0, 3, 5, 0)^T \textit{ representing } \{(v_1, v_3)\}.$$

Algorithm 6 Step 6 of Algorithm 5: Create a new temporary label.

Input: an instance $I = (E, Q, \hat{C}, \Delta, \boldsymbol{\Gamma})$, an edge $(v', v) \in E$, a label l' with cost $\boldsymbol{y'}$ at v'
Output: a new label l at v with predecessor label l'

```
1: for $i = 1, ..., k$ do
2:     Set $n_i := 1 + \sum_{j=1,...,(i-1)} (\Gamma_i + 1)$.
3:     $y_{n_i} := y'_{n_i} + \hat{c}_{(v',v),i}$
4:     $a := 1$
5:     while $a \leqq \Gamma_i$ do
6:         if $\delta_{(v',v),i} > y'_{n_i+a}$ then
7:             $y_{n_i+a} := \delta_{(v',v),i}$
8:             for $b := a+1, ..., \Gamma_i$ do $y_{n_i+b} := y'_{n_i+b-1}$
9:             end for
10:            $a := \Gamma_i + 1$
11:        else
12:            $y_{n_i+a} := y'_{n_i+a}$
13:            $a := a + 1$
14:        end if
15:    end while
16: end for
17: Create the temporary label $l := ((y_0, ..., y_m)^T, v', l')$ at node $v$.
```

We now have one permanent label l_0 and two temporary labels l_2^1, l_3^1. The aggregated cost of l_2^1 is smaller than the aggregated cost of l_3^1.

3. *Because of its smaller aggregated cost, l_2^1 is made permanent in the next iteration of Line 4. New labels are created:*

$$\begin{aligned} &l_3^2 \textit{ at } v_3 \textit{ with cost } (3,1,1,2,5,1)^T \textit{ representing } \{(v_1,v_2),(v_2,v_3)\} \\ &l_4^1 \textit{ at } v_4 \textit{ with cost } (5,4,1,5,1,1)^T \textit{ representing } \{(v_1,v_2),(v_2,v_4)\} \\ &l_5^1 \textit{ at } v_2 \textit{ with cost } (5,5,1,3,5,1)^T \textit{ representing } \{(v_1,v_2),(v_2,v_5)\}. \end{aligned}$$

As an example, we look at the creation of l_3^2 in detail: The cost vector of l_2^1 is $(2,1,0,1,1,0)^T =: \boldsymbol{y'}$. We obtain

$$\boldsymbol{y'} \oplus (\boldsymbol{\hat{c}_{(v_2,v_3)}}, \boldsymbol{\delta_{(v_2,v_3)}}) = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \oplus \left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 5 \end{pmatrix} \right) = \begin{pmatrix} y'_1 + \hat{c}_{(v_2,v_3),1} \\ y'_2 \\ \delta_{(v_2,v_3),1} \\ y'_4 + \hat{c}_{(v_2,v_3),2} \\ \delta_{(v_2,v_3),2} \\ y'_5 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 1 \\ 2 \\ 5 \\ 1 \end{pmatrix},$$

because $y'_2 \geqq \delta_{(v_2,v_3),1} > y'_3$ and $\delta_{(v_2,v_3),2} > y'_5$. The cost vectors of the two labels l_3^1, l_3^2 at v_3 are compared to each other. As none dominates the other, both are kept.
The labels l_0, l_2^1 are now permanent. We have four temporary labels $l_3^1, l_3^2, l_4^1, l_5^1$, among which l_3^2 has the smallest aggregated cost.

After several iterations of Lines 4 to 13, there do not exist any temporary labels. Algorithm 5 *returns 3 permanent labels at node v_6:*

one with cost $(8,4,1,7,1,1)^T$ *representing* $q_1 = \{(v_1,v_2),(v_2,v_4),(v_4,v_6)\}$,
one with cost $(6,3,2,6,5,5)^T$ *representing* $q_2 = \{(v_1,v_2),(v_2,v_3),(v_3,v_5),(v_5,v_6)\}$,
one with cost $(8,3,1,6,5,5)^T$ *representing* $q_3 = \{(v_1,v_2),(v_2,v_3),(v_3,v_4),(v_4,v_6)\}$.

In Algorithm 7 non-dominated paths according to their worst case cost will be identified from the obtained labels, see Example 29.

Lemma 24. *In Algorithm 5 for every label $l = (\boldsymbol{y}, v', l')$ at a node v there exists a path q from s to v with $\boldsymbol{y} = \boldsymbol{z}^{\mathbf{D}}(q)$.*

Proof. We show the statement by induction:
The first label has cost $(0,...,0)$ and represents the path only consisting of node s.
Let $\boldsymbol{y'} = (y'_1,...,y'_m)$ be the cost of the predecessor label l' and assume that $\boldsymbol{y'}$ equals the cost $\boldsymbol{z}^{\mathbf{D}}(q')$ of a path q' from s to v'. Let $q := q' \cup (v',v)$. Then we have

$$\forall i = 1,...,k : y_{n_i} = y'_{n_i} + \hat{c}_{(v',v),i} = \sum_{e\in q'} \hat{c}_{e,i} + \hat{c}_{(v',v),i} = \sum_{e\in q} \hat{c}_{e,i}.$$

Further, we distinguish two cases for all $i = 1,...,k$:

- Case 1: $\delta_{(v',v),i} \leqq y'_{n_i+a} \forall\ a = 1,...,\Gamma_i$. In this case the Γ_i edges e with biggest intervals $\delta_{e,i}$ of q' and $q' \cup (v',v)$ are the same and $y_{n_i+a} = y'_{n_i+a}$ for all $a = 1,...,\Gamma_i$. Therefore, $(y_{n_i},...,y_{n_i+\Gamma_i}) = (z^{\mathrm{D}}_{n_i}(q),...,z^{\mathrm{D}}_{n_i+\Gamma_i}(q))$.
- Case 2: Either $\delta_{(v',v),i} > y'_{n_i+a}$ for $a = 1$ or $\exists\ a \in \{2,...,\Gamma_i\}$ with $y'_{n_i+a-1} \geqq \delta_{(v',v),i} > y'_{n_i+a}$. Then

$$\begin{aligned}
&\forall\ b < a : && y_{n_i+b} = y'_{n_i+b} \text{ and } b\text{-}\max_{e\in q'} \delta_{e,i} = b\text{-}\max_{e\in q'\cup(v',v)} \delta_{e,i}\\
&\text{for } \ b = a : && y_{n_i+b} = \delta_{(v',v),i} = b\text{-}\max_{e\in q'\cup(v',v)} \delta_{e,i}\\
&\forall\ b : \Gamma_i \geqq b > a : && y_{n_i+b} = y'_{n_i+b-1} = b\text{-}\max_{e\in q\cup(v',v)} \delta_{e,i}
\end{aligned}$$

It follows $(y_1,...,y_m) = \boldsymbol{z}^{\mathbf{D}}(q)$. □

In the deterministic case with only sum functions, subpaths of efficient paths are efficient as well, which plays an important role in the proof of Martin's algorithm. If some of the objective functions are bottleneck functions, this property does not hold any more (Gandibleux et al., 2006). In our case, since we only look for a complete set of efficient solutions, the weaker property given in Lemma 26 is sufficient (this was observed but not proven by Iori et al. (2010)).
We use the following notation to specify subpaths.

Notation 25. *Let q be a simple path and v, w two nodes on q (v before w). Let then $q_{v,w}$ denote the part of q from node v to node w.*

Lemma 26. *Let q from s to t be an efficient path with respect to $\boldsymbol{z}^{\boldsymbol{D}}$ and v, w two nodes on q (v before w). Then either $q_{v,w}$ is an efficient path from v to w or there exists an efficient path p from v to w such that $q' := q_{s,v} \cup p \cup q_{w,t}$ is equivalent to q.*

Proof. Assume that $q_{v,w}$ is not efficient w.r.t $\boldsymbol{z}^{\mathbf{D}}$. Then there exists an efficient path p from v to w that dominates $q_{v,w}$. We have

$$\sum_{e\in q'}\hat{\boldsymbol{c}}_{\boldsymbol{e}} = \sum_{e\in q_{s,v}}\hat{\boldsymbol{c}}_{\boldsymbol{e}} + \sum_{e\in p}\hat{\boldsymbol{c}}_{\boldsymbol{e}} + \sum_{e\in q_{w,t}}\hat{\boldsymbol{c}}_{\boldsymbol{e}} \leqq \sum_{q_{s,v}}\hat{\boldsymbol{c}}_{\boldsymbol{e}} + \sum_{e\in q_{v,w}}\hat{\boldsymbol{c}}_{\boldsymbol{e}} + \sum_{e\in q_{w,t}}\hat{\boldsymbol{c}}_{\boldsymbol{e}} = \sum_{e\in q}\hat{\boldsymbol{c}}_{\boldsymbol{e}}.$$

As p dominates $q_{v,w}$, it follows $\forall\ i = 1,...,k, a = 1,...,\Gamma_i$: $a\text{-}\max_{e\in p}\delta_{e,i} \leqq a\text{-}\max_{e\in q_{v,w}}\delta_{e,i}$, and hence $a\text{-}\max_{e\in q'}\delta_{e,i} \leqq a\text{-}\max_{e\in q}\delta_{e,i}\ \forall\ i = 1,...,k, a = 1,...,\Gamma_i$.
It follows $\boldsymbol{z}^{\mathbf{D}}(q') \leqq \boldsymbol{z}^{\mathbf{D}}(q)$ and we conclude $\boldsymbol{z}^{\mathbf{D}}(q') = \boldsymbol{z}^{\mathbf{D}}(q)$, because q is efficient with respect to $\boldsymbol{z}^{\mathbf{D}}$. □

Theorem 27. *When Algorithm 5 (with Algorithm 6 as Step 6) stops, the permanent labels at t represent a complete set of efficient solutions for (MODCO).*

Proof. We have to show that each permanent label at t represents an efficient path from s to t and that for each efficient path q from s to t a permanent label at t representing q or an equivalent path exists.
The proof of the first part is analogous to the proof by Ehrgott (2006) of the multi-objective label setting algorithm by Martins (1984). For substituting the lexicographic order with the aggregate cost order we refer to Iori et al. (2010).
Now, we show that for each efficient path q from s to t a permanent label at t representing q or an equivalent path exists. Assume that we have an efficient path q from s to t, such that there is no permanent label l at t with label costs $\boldsymbol{y} = \boldsymbol{z}^{\mathbf{D}}(q)$. Consider the predecessor node v' of t on q. From Lemma 26 it follows that there is an efficient path p from s to v' with $\boldsymbol{z}^{\mathbf{D}}(p \cup (v',t)) = \boldsymbol{z}^{\mathbf{D}}(q)$.
If there exists a permanent label l' at v' with label costs $\boldsymbol{y'} = \boldsymbol{z}^{\mathbf{D}}(p)$, then, after it was made permanent in Line 4, a new label $\bar{l}$ at node t with label costs $\bar{\boldsymbol{y}} = \boldsymbol{y'} \oplus (\hat{\boldsymbol{c}}_{(\boldsymbol{v'},\boldsymbol{t})}, \boldsymbol{\delta}_{(\boldsymbol{v'},\boldsymbol{t})})$ was constructed in Line 6. It follows

$$\bar{\boldsymbol{y}} = \boldsymbol{y}' \oplus (\hat{\boldsymbol{c}}_{(\boldsymbol{v'},\boldsymbol{t})}, \boldsymbol{\delta}_{(\boldsymbol{v'},\boldsymbol{t})}) = \boldsymbol{z}^{\mathbf{D}}(p) \oplus (\hat{\boldsymbol{c}}_{(\boldsymbol{v'},\boldsymbol{t})}, \boldsymbol{\delta}_{(\boldsymbol{v'},\boldsymbol{t})}) = \boldsymbol{z}^{\mathbf{D}}(p \cup (v',t)) = \boldsymbol{z}^{\mathbf{D}}(q).$$

Consider the first label with cost $\boldsymbol{z}^{\mathbf{D}}(q)$ that was constructed at node t. If this label was deleted again, its cost vector is dominated, which contradicts the efficiency of q. If it was not deleted, then it was made permanent, which contradicts our assumption that no permanent label with costs $\boldsymbol{z}^{\mathbf{D}}(q)$ exists at t.
Therefore, there is no permanent label at the predecessor node v' of t with costs $\boldsymbol{y'}$ such that $\boldsymbol{y'} \oplus (\hat{\boldsymbol{c}}_{\boldsymbol{e}}, \boldsymbol{\delta}_{\boldsymbol{e}}) = \boldsymbol{z}^{\mathbf{D}}(q)$. In the same way, we can show that there is no permanent label at the predecessor node v'' of v' with costs $\boldsymbol{y''}$ such that

$$\left(\boldsymbol{y''} \oplus (\hat{\boldsymbol{c}}_{(\boldsymbol{v''},\boldsymbol{v'})}, \boldsymbol{\delta}_{(\boldsymbol{v''},\boldsymbol{v'})})\right) \oplus (\hat{\boldsymbol{c}}_{(\boldsymbol{v'},\boldsymbol{t})}, \boldsymbol{\delta}_{(\boldsymbol{v'},\boldsymbol{t})}) = \boldsymbol{y'} \oplus (\hat{\boldsymbol{c}}_{(\boldsymbol{v'},\boldsymbol{t})}, \boldsymbol{\delta}_{(\boldsymbol{v'},\boldsymbol{t})}) = \boldsymbol{z}^{\mathbf{D}}(q).$$

By induction it follows that there is no permanent label at node s with cost $(0,...,0)$, which is a contradiction, because such a label is constructed in Line 2 of the algorithm and made permanent during the first execution of Line 4.
We conclude that for each efficient path q from s to t there exists a permanent label at t representing q or a path that is equivalent to q. Furthermore, each permanent label at t represents an efficient path from s to t. Therefore, the paths represented by the permanent labels are a complete set of efficient solutions. □

Algorithm 7 LSA for the shortest path problem with cardinality-constrained uncertainty

```
Input: an instance $I = (E, Q, \hat{C}, \Delta, \mathbf{\Gamma})$ of (MOUSP)
Output: a complete set of robust efficient solutions for $I$
1: Solve (MODCO) with Algorithm 5.
2: For every permanent label $l$ in $t$ compute the worst case costs $\boldsymbol{z}^{\mathbf{R}}(q)$ of its represented
   path $q$ by $z_i^{\mathrm{R}}(q) := \sum_{i=n_i,\dots,n_i+\Gamma_i} y_i$ and choose the non-dominated ones.
3: Obtain the represented paths by backtracking the predecessor labels.
```

To find a a complete set of robust efficient solutions we have to filter the solutions obtained by the labeling algorithm (see Algorithm 7).

Corollary 28. *Algorithm 7 finds a complete set of robust efficient solutions for an instance $I = (E, Q, \hat{C}, \Delta, \mathbf{\Gamma})$ of (MOUSP) with $\hat{C}$ being entry-wise non-negative.*

Example 29. *Consider the instance given in Example 7. From the permanent labels returned by Algorithm 5 (see Example 23), the worst costs of their represented paths are computed:*

$$\begin{aligned}
\boldsymbol{z}^{\boldsymbol{R}}(q_1) &= (8+4+1, 7+1+1)^T = (13, 9)^T \\
\boldsymbol{z}^{\boldsymbol{R}}(q_2) &= (6+3+2, 6+5+5)^T = (11, 16)^T \\
\boldsymbol{z}^{\boldsymbol{R}}(q_3) &= (8+3+1, 6+5+5)^T = (12, 16)^T.
\end{aligned}$$

Since $\boldsymbol{z}^{\boldsymbol{R}}(q_3)$ is dominated by $\boldsymbol{z}^{\boldsymbol{R}}(q_2)$, only the paths q_1 and q_2 are returned by Algorithm 7.

4. Experimental evaluation

In this paper we presented two approaches to find a complete set of robust efficient solutions for (MOUCO). DSA solves the uncertain problem, assuming that we know how to solve the deterministic multi-objective problem. To use the bottleneck approach we need a method to solve a deterministic multi-objective problem with several objective functions, some of which are sums and some of which are bottleneck functions. We introduced such an algorithm for the shortest path problem (LSA) and, hence, we test our approaches on the shortest path problem (MOUSP).

4.1. Hazardous material transportation

We test our algorithms for (MOUSP) on a hazardous material transportation instance: When transporting hazardous materials, on one hand, the shipping company wants to minimize travel time, distance or fuel costs. On the other hand, if an accident happens, environment and population are exposed to the hazardous material. Hence, another objective is to keep the risk and negative impacts of accidents to a minimum. Erkut et al. (2007) give an overview about objectives for hazardous material transportation and about approaches for estimating the risk and the impacts of an accident.
For our experiments we consider the travel time and the population affected by a potential accident. We assume a nominal travel time on each road and a potential delay resulting from congestion or incidents like accidents or road construction works on some of the roads. We further assume a nominal population level, which can be increased locally by events like fairs or sport events, or due to regular shifts in population during the workday.

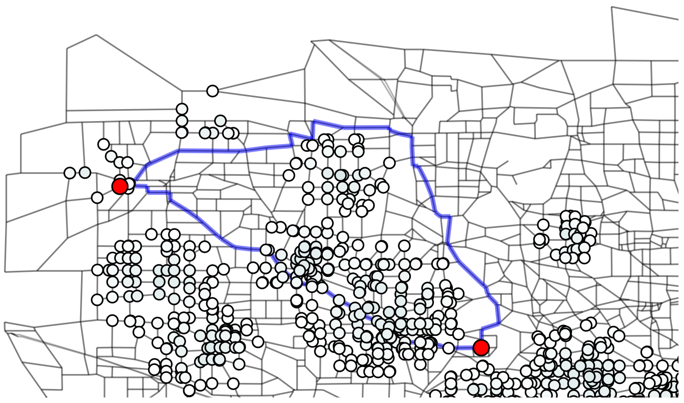


Figure 3: Section of the Chicago regional road network with distribution of population (see Kuhn et al., 2016). The red dots show start and end node chosen for our experiments and two exemplary robust efficient paths are marked in blue.

Our problem instance for hazardous material transportation is based on the instance used by Kuhn et al. (2016) to test an algorithm for bi-objective shortest path problems with only one uncertain objective. The underlying network (Chicago-regional) is a sector of the Chicago region road network available from Bar-Gera et al.. The sector contains 1301 nodes and 4091 edges.

To obtain plausible travel times, Kuhn et al. (2016) solve a traffic assignment problem with an iterative algorithm. It models the simultaneous movement of network users, assuming travelers follow their shortest paths. Congestion effects are taken into account by a nonlinear relationship between the flow on an edge and the travel time. Until an equilibrium solution is found, each iteration of the algorithm produces a flow and resulting travel times on the edges. To obtain the lower (upper) limit of the travel time interval for each edge we choose the smallest and largest travel times obtained during several stages of the iterative equilibrium algorithm.

For the population we use the distribution of the population described by Kuhn et al. (2016) as nominal values (lower interval limits). We randomly assign integer interval lengths ($\delta_{e,2}$) up to x% of the respective nominal value. By varying x we obtain several test instances. We call x the *population uncertainty*.

We choose an appropriate start and end node with an agglomeration of population between them. Figure 3 shows two exemplary robust efficient paths for the instance with $x = 10$ and $\mathbf{\Gamma} = (5, 5)$. One of the paths goes directly through the area with high population. Here the time objective function has a small value, whereas the number of people exposed to the risk of health damage in case of an accident is relatively high. The other path avoids highly populated areas, which results in a longer travel time.

4.2. Results

The algorithms are implemented in C++, compiled under Debian 8.6 with g++ 4.9.2 compiler, and run on a Laptop with 2.10 GHz quad core processor and 7.71 GB of RAM. If not stated otherwise, we use an implementation of DSA that contains all enhancements described in Section 3.1. In addition, it checks in the beginning, whether the instance has objective-independent element order. If this is the case, we use a special version of DSA, as proposed in Lemma 17, which we will refer to as DSA-oi: Instead of the nested for-loops in Lines 3 to

6 of Algorithm 4 it only contains one for-loop.
For solving the subproblems we use an implementation of the algorithm by Martins (1984) (with the difference that the labels are selected w.r.t. their aggregate cost instead of using the lexicographic order). There and in the implementation of LSA, we additionally delete new labels at any node if they are dominated by an existing label at t.
In the figures, one data point represents one measurement, except for Section 4.2.3, where we took the average running time of 40 runs.
To compare the performance of our solution approaches, we solve the bi-objective hazardous material transportation instance described above for different values of population uncertainty x and Γ. We always choose the same value for Γ_1 and Γ_2 and we will refer to this value as Γ_i in the following. In addition, we compare the performance of the algorithms on an instance with with two correlated objective functions and on an instance with three objectives. We further evaluate the improvement gained by our enhancement of DSA (solution checking). Finally, we generate an instance with objective-independent element order and investigate to which extent the performance time of the DSA benefits from the results in Lemma 17.

4.2.1. Number of robust efficient solutions for the hazardous material transportation instance

Figure 4 shows the minimal cardinality of a complete set of robust efficient solutions for the generated instances for several values of x and Γ_i. In general, for increasing values of population uncertainty x the number of robust efficient solutions increases as well, because of the higher variation allowed in the second objective. We do not observe a direct dependency on Γ_i, but for values greater than 25 the number of robust efficient solutions stays the same or differs only little. The reason is that the robust efficient solutions contain only between 39 and 56 edges. Furthermore, the interval lengths $\delta_{e,1}$ resp. $\delta_{e,2}$ of some edges are 0. Hence, at some point, allowing more edges to differ from their minimal cost makes no difference.
In Table 1 we present the number of solutions generated in total: For DSA we add the number of solutions obtained by solving the subproblems (which possibly contain identical solutions several times). For LSA we list the number of solutions found by the multi-objective labeling algorithm before the filtering step. The number of solutions generated increases with the population uncertainty x (as does the number of robust efficient solutions). It tends to decrease for increasing Γ_i (with a few exceptions). For the DSA that is because of the decreasing number of subproblems solved (see Figure 8(b)).

4.2.2. Comparison of the two solution approaches

Figure 5 shows the running time of DSA and LSA for several values of Γ_i and x. The running time of LSA increases with Γ_i, whereas the running time of DSA decreases (see also Figure 8(a)). The reason is that for increasing Γ_i, the number of objectives in the deterministic multi-objective problem solved during LSA increases as well. However, the maximal number of subproblems solved during DSA decreases. For small values of Γ_i LSA solves the given instances faster, for higher values DSA has a better performance.
Choosing a higher value for x results in a greater maximal and mean deviation from the nominal value and a higher number of different values of $\delta_{e,2}$. When x is increased, the running time of both algorithms increases. In the case of DSA, this can be explained by the higher number of different values of $\delta_{e,2}$, which leads to a higher number of subproblems.

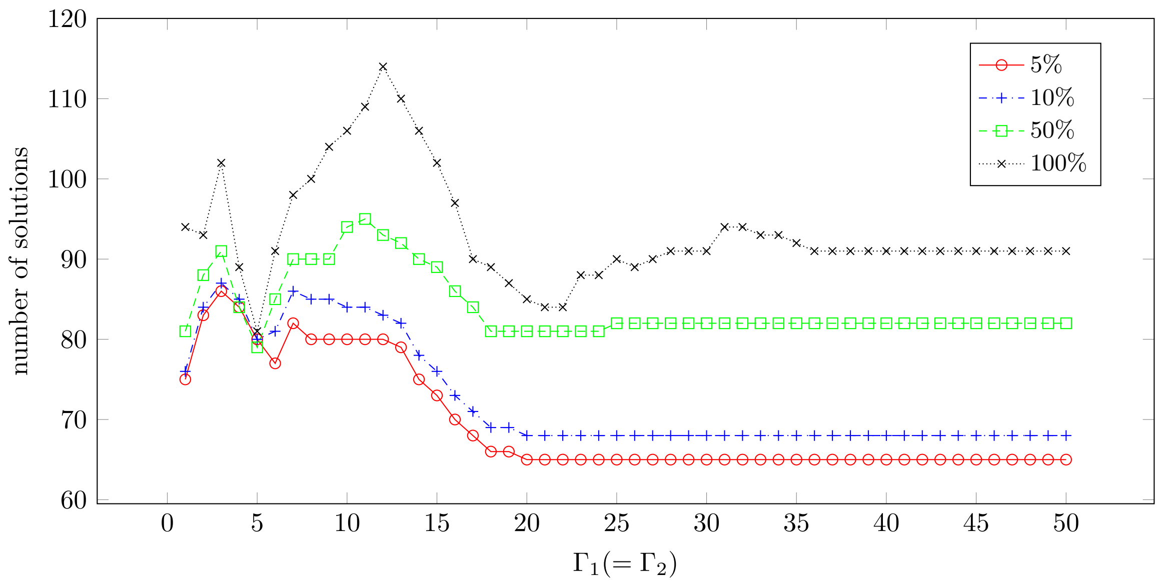


Figure 4: Number of robust efficient solutions for several values of Γ_i and population uncertainty x.

	pop. unc. $\leqq$ 5%			pop. unc. $\leqq$ 10%			pop. unc. $\leqq$ 50%			pop. unc. $\leqq$ 100%		
Γ_i	sol	tDSA	tLSA	sol	tDSA	tLSA	sol	tDSA	tLSA	sol	tDSA	tLSA
1	75	26288	6991	76	52887	8886	81	226008	13189	94	468828	16768
2	83	26278	4529	84	52867	5879	88	225928	7830	93	468668	10228
3	86	26579	2972	87	53544	3732	91	229031	4727	102	475140	5860
4	84	26569	1679	85	53524	2057	84	228951	2184	89	474980	2843
5	80	26569	691	80	53524	944	79	228951	843	81	474980	940
10	80	25179	–	84	50665	–	94	216596	–	106	449430	–
20	65	23306	–	68	46912	–	81	200709	–	85	407281	–
30	65	21762	–	68	39437	–	82	178838	–	91	367987	–
40	65	20264	–	68	32655	–	82	154478	–	91	330851	–
50	65	15011	–	68	30306	–	82	135934	–	91	296009	–

Table 1: Number of generated solutions for several values of Γ_i and population uncertainty: sol = minimal number of robust efficient solutions in a complete set, tDSA = total number of solutions generated in the subproblems, tLSA= total number of solutions found with the multi-objective labeling algorithm (before filtering the robust efficient solutions).

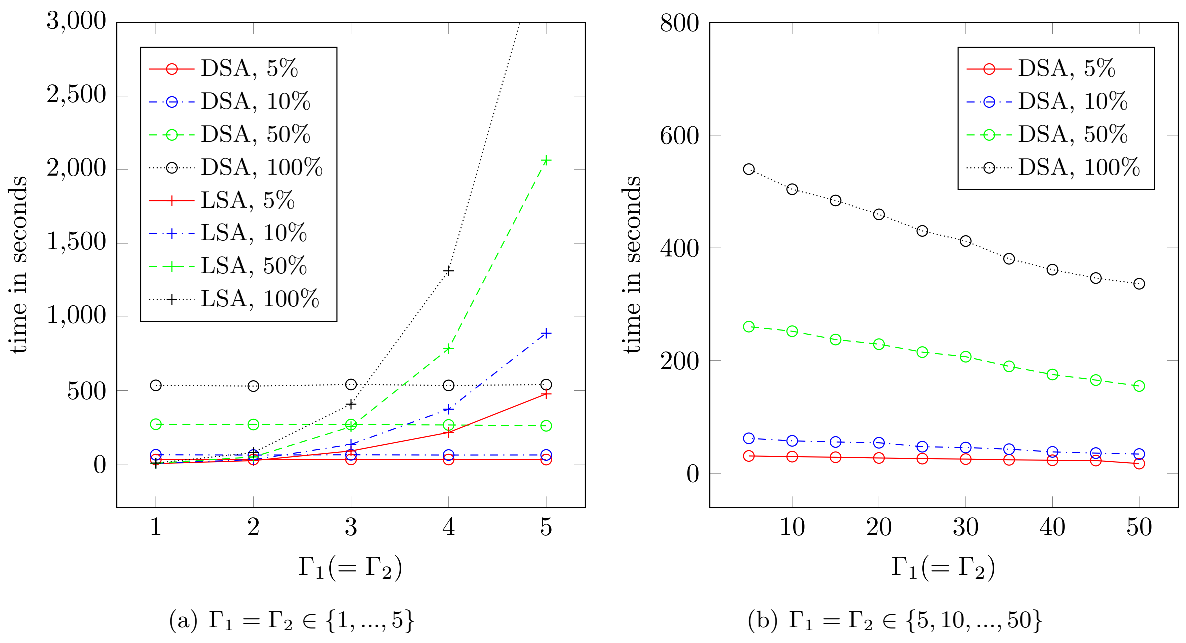


Figure 5: Running time of DSA and LSA for several values of Γ_i and population uncertainty x on two different scales.

4.2.3. Correlated objective functions

We additionally generate an instance with two strongly correlated objective functions: We use the travel time as one objective and generate a second travel time objective by multiplying the nominal times and the interval lengths each by a random factor between 0.9 and 1.1.
Both algorithms benefit a lot from the correlation, all running times are now less than four seconds, as shown in Figure 6. In comparison, LSA benefits more from correlated objective function values: The values of Γ_i, for which it is still faster than DSA, are much higher on this instance than on the original hazardous material transportation instance considered in Section 4.2.2. For small values of Γ_i it is much faster than DSA.

4.2.4. Three objectives

Since we are also interested in the performance of the algorithms for problems with more than two objectives, we generate an artificial third objective: For the nominal values we use again the nominal population. We generate random interval lengths in the same range as the other population objective. That means, the value of population uncertainty in general is the same for both population objectives, but the specific interval lengths of each edge may differ. Because of the identical nominal values, two of the three objectives are correlated. Figure 7 shows the running times on this instance in comparison to the instance with two objectives described above.
The running time of both algorithms increases by including the additional objective, even though it is strongly correlated to one of the original objectives. The relative difference between the running time of the instance with two objectives and the instance with three objectives increases with Γ_i for LSA, whereas it decreases for DSA.

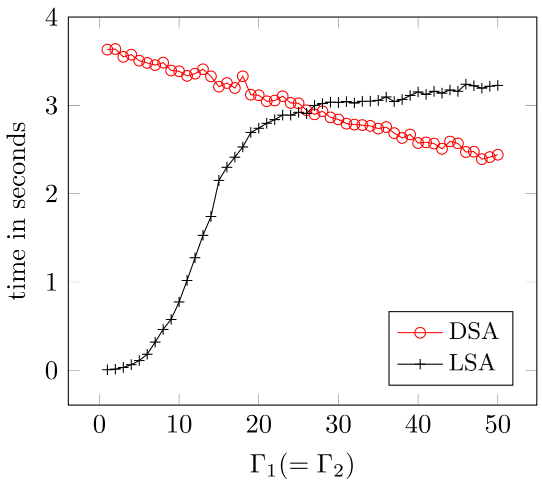
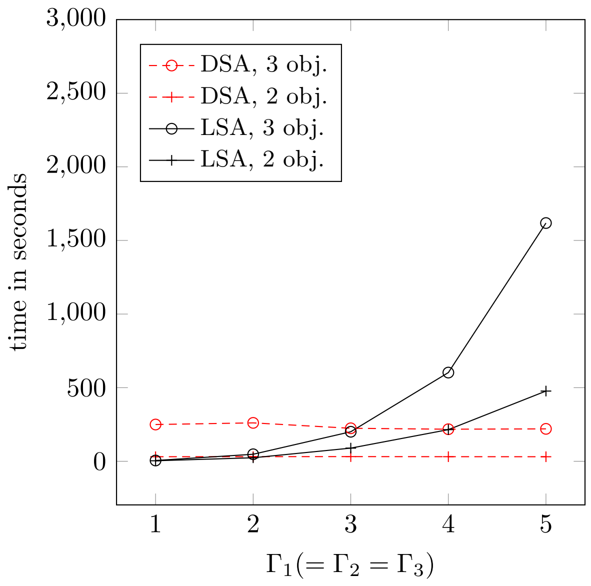
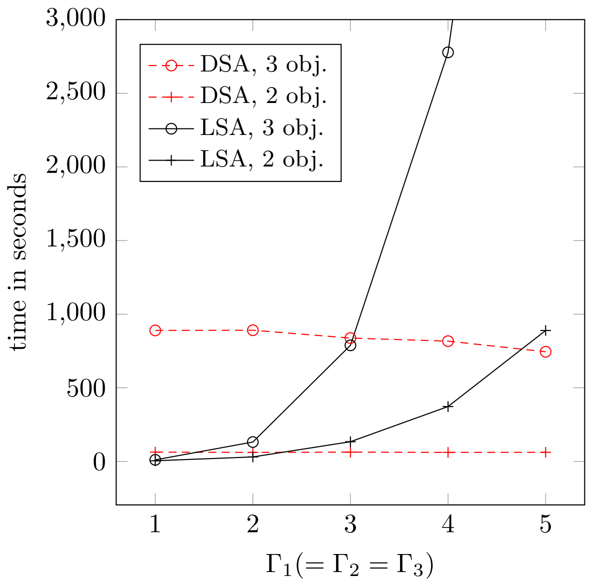


Figure 6: Running time of DSA and LSA for an instance with two strongly correlated objective functions.

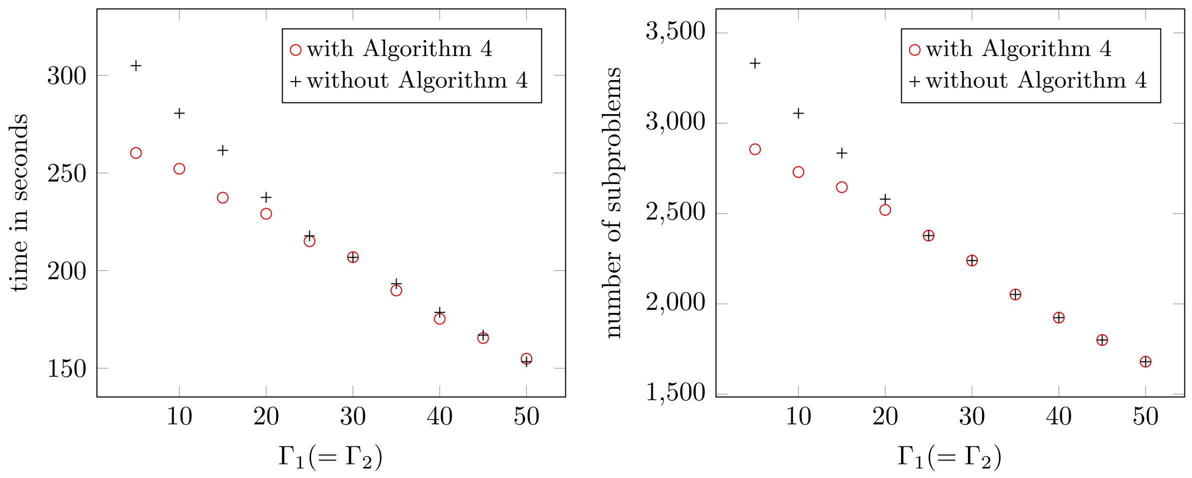


(a) Population uncertainty 5%



(b) Population uncertainty 10%

Figure 7: Running time of DSA and LSA for an instance with three objectives and an instance with two objectives.



(a) Running time.

(b) Number of Subproblems.

Figure 8: Running time and number of solved subproblems of DSA with and without solution checking (Population uncertainty 50%).

4.2.5. Evaluation of the improvement obtained by solution checking

To evaluate the obtained improvement by using solution checking in DSA, we use Algorithm 4 as Step 3 of Algorithm 3. We compare the running time of the version containing solution checking to the running time of the version without this enhancement (Figure 8(a)). Additionally, we count the solved subproblems (Figure 8(b)). Where fewer subproblems were solved because of the enhancement, the running times differ significantly, for all other instances they are nearly equal. Hence, the check itself does not slow down the algorithm significantly in comparison to the acceleration that we obtain when subproblems can be skipped. We conclude that it is worth using the enhancement, but as Γ_i increases solution checking becomes less effective.
Note that, since Lemma 13 allows to exclude even more subproblems than excluded in Algorithm 4, further speed-ups may be achieved by implementing a more sophisticated solution checking. However, already when using Algorithm 4, the benefit of solution checking is clearly visible.

4.2.6. Evaluation of DSA for instances with objective-independent element order

For instances with objective-independent element order, we use the special version DSA-oi as proposed in Lemma 17. To compare its performance to the general version of DSA we construct an instance with objective-independent element order: Instead of generating interval lengths for the population objective we use the interval lengths of the travel time objective. Figure 9 shows that DSA-oi has a much better performance than the general algorithm. The test, whether the instance is objective-independent, only takes a small fraction of the running time (for our instances $1.4 \cdot 10^{-5}$ seconds). Therefore, it is reasonable to check each instance for objective-independent element order before solving it with DSA.

5. Conclusion

In this paper we developed two approaches to find minmax robust solutions for multi-objective combinatorial optimization problems with cardinality-constrained uncertainty. We extended

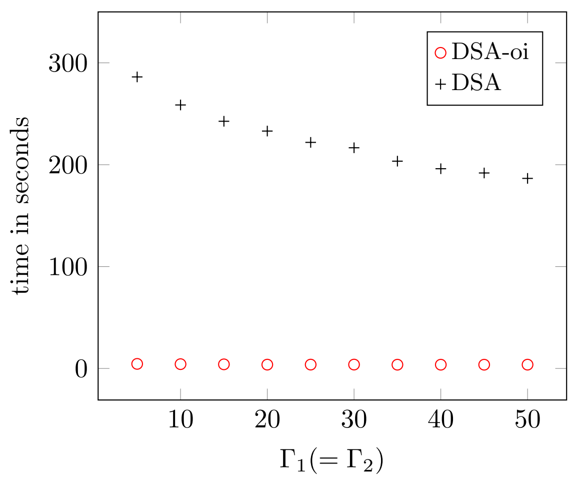


Figure 9: Comparison of DSA and DSA-oi for instances with objective-independent element order.

an algorithm by Bertsimas and Sim (2003) to multi-objective optimization (DSA), suggested an enhancement and developed a special version for instances with objective-independent element order. We also introduced a second approach and used it to develop a label setting algorithm (LSA) for the multi-objective uncertain shortest path problem.
We tested our algorithms on several instances of the multi-objective uncertain shortest path problem arising from hazardous material transportation. On most of the tested instances DSA has a better performance, but LSA is faster for small values of Γ_i. If the two objective functions are strongly correlated, LSA is competitive even for higher values of Γ_i. This appears often in shortest path problems, where, e.g., the distance, travel time and fuel consumption are correlated.
When implementing DSA we recommend to use the proposed enhancements and to check whether the special version for instances with (partial) objective-independent element order can be used. The checks do not take long in comparison to the total running time, and if their result is positive, the algorithm can be accelerated significantly.
For further investigations other variants of multi-objective cardinality-constrained uncertainty are of interest. A second way to extend the single-objective concept is to require the edges whose costs differ from their minimal values to be the same for all objectives. In this case the uncertainties in the objectives are no longer independent of each other and using point-based or set-based minmax robust efficiency leads to different solution sets. An interesting variation of cardinality-constrained uncertainty is not to consider a bound on the cardinality, but on the sum of the deviation from their minimal values.
Further research on robust multi-objective optimization includes other types of uncertainty, e.g., discrete scenario sets or polyhedral or ellipsoidal uncertainty. Also the case of decision uncertainty, in which the solution found cannot be realized exactly, is of interest, see (Eichfelder et al., 2017) for first results.
The algorithms for the multi-objective cardinality-constrained uncertain shortest path problem presented in this paper can easily be extended to the *multi-objective single-source shortest path problem*. There, a complete set of efficient paths from a start node s to all other nodes is to be found. In the deterministic case, there exist algorithms (e.g. the algorithm by Martins (1984)) for which it can be shown that the running time is polynomial in the output size. It

would be interesting to investigate whether this is the case for the uncertain problem, too.

Acknowledgments

Lisa Thom was supported by DFG RTG 1703 “Resource Efficiency in Interorganizational Networks”.

References

Aissi, H., Bazgan, C., and Vanderpooten, D. (2009). Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438.

Bar-Gera, H., Kwon, C., Li, J., and Stabler, B. Transportation networks. `https://github.com/bstabler/TransportationNetworks`. Accessed: 2016-11-04.

Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust optimization.* Princeton University Press.

Bertsimas, D. and Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical programming*, 98(1):49–71.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations research*, 52(1):35–53.

Chuong, T. D. (2016). Optimality and duality for robust multiobjective optimization problems. *Nonlinear Analysis: Theory, Methods & Applications*, 134:127–143.

Doolittle, E., Kerivin, H. M., and Wiecek, M. M. (2012). A robust multiobjective optimization problem with application to internet routing. Technical Report R2012-11-DKW, Clemson University.

Ehrgott, M. (2006). *Multicriteria optimization.* Springer, Berlin, Heidelberg.

Ehrgott, M., Ide, J., and Schöbel, A. (2014). Minmax robustness for multi-objective optimization problems. *European Journal of Operational Research*, 239:17–31.

Eichfelder, G., Krüger, C., and Schöbel, A. (2017). Decision uncertainty in multiobjective optimization. *Journal of Global Optimization*, pages 1–26. online first.

Erkut, E., Tjandra, S. A., and Verter, V. (2007). Hazardous materials transportation. *Handbooks in operations research and management science*, 14:539–621.

Fliege, J. and Werner, R. (2014). Robust multiobjective optimization & applications in portfolio optimization. *European Journal of Operational Research*, 234(2):422–433.

Gandibleux, X., Beugnies, F., and Randriamasy, S. (2006). Martins’ algorithm revisited for multi-objective shortest path problems with a maxmin cost function. *4OR*, 4(1):47–59.

Gorski, J., Klamroth, K., and Ruzika, S. (2012). Generalized multiple objective bottleneck problems. *Operations Research Letters*, 40(4):276–281.

Hassanzadeh, F., Nemati, H., and Sun, M. (2013). Robust optimization for multiobjective programming problems with imprecise information. *Procedia Computer Science*, 17:357 – 364.

Ide, J., Köbis, E., Kuroiwa, D., Schöbel, A., and Tammer, C. (2014). The relationship between multi-objective robustness concepts and set valued optimization. *Fixed Point Theory and Applications*, 2014(83).

Ide, J. and Schöbel, A. (2016). Robustness for uncertain multi-objective optimization: A survey and analysis of different concepts. *OR Spectrum*, 38(1):235–271.

Ide, J., Tiedemann, M., Westphal, S., and Haiduk, F. (2015). An application of deterministic and robust optimization in the wood cutting industry. *4OR*, 13(1):35–57.

Iori, M., Martello, S., and Pretolani, D. (2010). An aggregate label setting policy for the multi-objective shortest path problem. *European Journal of Operational Research*, 207(3):1489–1496.

Kalantari, M., Dong, C., and Davies, I. J. (2016). Multi-objective robust optimisation of unidirectional carbon/glass fibre reinforced hybrid composites under flexural loading. *Composite Structures*, 138:264–275.

Kuhn, K., Raith, A., Schmidt, M., and Schöbel, A. (2016). Bicriteria robust optimization. *European Journal of Operational Research*, 252:418–431.

Kuroiwa, D. and Lee, G. M. (2012). On robust multiobjective optimization. *Vietnam J. Math*, 40(2-3):305–317.

Lee, T. and Kwon, C. (2014). A short note on the robust combinatorial optimization problems with cardinality constrained uncertainty. *4OR*, 12(4):373–378.

Martins, E. Q. V. (1984). On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236–245.

Park, K.-C. and Lee, K.-S. (2007). A note on robust combinatorial optimization problem. *Management Science and Financial Engineering*, 13(1):115–119.

Soyster, A. L. (1973). Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations research*, 21(5):1154–1157.

Wiecek, M. M. and Dranichak, G. M. (2016). Robust multiobjective optimization for decision making under uncertainty and conflict. In *Optimization Challenges in Complex, Networked and Risky Systems*, pages 84–114. INFORMS.

Yu, H. and Liu, H. M. (2013). Robust multiple objective game theory. *Journal of Optimization Theory and Applications*, 159(1):272–280.