

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

Andrea Raith<sup>a</sup>, Marie Schmidt<sup>b</sup>, Anita Schöbel<sup>c</sup>, and Lisa Thom<sup>c,\*</sup>

<sup>a</sup>Department of Engineering Science, The University of Auckland, postal address: Private Bag 92019, Auckland 1142, New Zealand, email address: a.raith@auckland.ac.nz

<sup>b</sup>Department of Technology and Operations Management, Rotterdam School of Management, Erasmus University Rotterdam, postal address: PO Box 1738, 3000 DR Rotterdam, The Netherlands, email address: schmidt2@rsm.nl

<sup>c</sup>Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen, postal address: Lotzestr. 16-18, 37083 Göttingen, Germany, email addresses:

schoebel@math.uni-goettingen.de (Anita Schöbel), l.thom@math.uni-goettingen.de (Lisa Thom)

\*Corresponding author, email address: l.thom@math.uni-goettingen.de

## 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 algorithm of Bertsimas and Sim (2003) 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 Multi-objective robust combinatorial optimization

### 1.1 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 in [ABV09]. Based on the interval case, Bertsimas and Sim propose in [BS04] 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. In [BS03] an algorithm is provided to find robust solutions for combinatorial optimization problems under this kind of uncertainty.

Only recently have robust optimization concepts for multi-objective problems been developed. A first extension of minmax robustness for several objectives was introduced in [KL12] and [FW14]. They consider the uncertainties in the objectives independently of each other. Ehrgott et al. developed another extension of minmax robustness [EIS14], in which they include the dependencies between the objectives, and which was generalized in [IKK<sup>+</sup>14]. These concepts have been extensively applied, e.g., in portfolio management [FW14], in game theory [YL13] and in the wood industry [ITWH15]. An overview on multi-objective robustness, including further robustness concepts, is given in [IS16] and [WD16]. Newest developments in this field include [Chu16] and [KDD16]. Cardinality constrained uncertainty has been extended to multi-objective optimization in [DKW12] (only for uncertain constraints) and [HNS13] (for uncertain objective functions and constraints).

To the best of our knowledge, only Kuhn et al. have developed a solution algorithm for multi-objective uncertain combinatorial optimization problems [KRSS16]. 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 arbitrarily many objectives of which all may be uncertain. The main contributions of this paper are that we develop two solution approaches for multi-objective combinatorial optimization problems with cardinality-constrained uncertainty and derive specific algorithms for the multi-objective uncertain shortest path problem.

The remainder of this paper is structured as follows: In Section 1 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 2: In Section 2.1 we extend an algorithm from [BS03] to multi-objective optimization and, additionally, 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 2.2 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 3, we compare our methods on instances of the multi-objective uncertain shortest path problem originating from hazardous material transportation.

## 1.2 Multi-objective optimization

First, we will give a short introduction to multi-objective optimization.

**Definition 1.** *Given a set  $\mathcal{X}$  of feasible solutions and  $k$  objective functions  $z_1, \dots, z_k : \mathcal{X} \rightarrow \mathbb{R}$  with  $k \geq 2$ , we call*

$$\min_{x \in \mathcal{X}} z(x) = \begin{pmatrix} z_1(x) \\ \vdots \\ z_k(x) \end{pmatrix}$$

*a multi-objective optimization problem (MOP).*

A solution that minimizes all objectives simultaneously does usually not exist. Therefore, we use the concept of *efficient solutions*.

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

$$\begin{aligned} y^1 \leq y^2 &\Leftrightarrow y_i^1 \leq y_i^2 \text{ for } i = 1, \dots, k \text{ and } y^1 \neq y^2, \\ y^1 \leq y^2 &\Leftrightarrow y_i^1 \leq y_i^2 \text{ for } i = 1, \dots, k. \end{aligned}$$

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

**Definition 3.** A solution  $x' \in \mathcal{X}$  dominates another solution  $x \in \mathcal{X}$  if  $z(x') \leq z(x)$ . We also say that  $z(x')$  dominates  $z(x)$ . A solution  $x \in \mathcal{X}$  is an efficient solution, if there is no  $x' \in \mathcal{X}$  such that  $x'$  dominates  $x$ . Then  $z(x)$  is called non-dominated.

Solving a multi-objective optimization problem  $\min\{z(x) = (z_1(x), \dots, z_k(x)) : x \in \mathcal{X}\}$  means to find its efficient solutions.

**Definition 4.** Two efficient solutions  $x, x' \in \mathcal{X}$  are called equivalent if  $z(x) = z(x')$ . A set of efficient solutions  $\bar{\mathcal{X}} \subseteq \mathcal{X}$  is called complete if all  $x \in \mathcal{X} \setminus \bar{\mathcal{X}}$  are either dominated by or equivalent to at least one  $x' \in \bar{\mathcal{X}}$ .

### 1.3 Robust optimization

We briefly introduce robust optimization for single-objective problems.

In robust optimization the uncertain input data is given as an *uncertainty set*  $\mathcal{U}$ , containing all possible *scenarios* that can occur. For each scenario  $\xi \in \mathcal{U}$  we obtain a different instance of the optimization problem  $\min_{x \in \mathcal{X}} z(x, \xi)$ .

**Definition 5.** Given a feasible set of solutions  $\mathcal{X}$ , an uncertainty set  $\mathcal{U}$  and an objective function  $z : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ , we define an uncertain optimization problem as the family of parameterized problems on  $\mathcal{X}$

$$\left( \min_{x \in \mathcal{X}} z(x, \xi), \xi \in \mathcal{U} \right).$$

We only consider problems with uncertainty in the objective function, not in the constraints. This is because, in the considered robustness concepts, a solution is only robust feasible if it is feasible for every scenario. This is reasonable for many combinatorial optimization problems, e.g., when choosing a path in a road or transportation network: If we decide on a path to take without knowing which scenario will occur, this path should at least exist for every scenario. Hence, we have deterministic constraints.

There are different *robustness concepts* offering a definition of a *robust solution* for an uncertain optimization problem, usually by defining a deterministic problem, called the *robust counterpart* (see [GS16] for an overview). The concept of *minmax robustness*, also called *strict* or *worst case* robustness, seeks solutions, for which the worst possible objective value is minimized. The solutions can be found by solving the robust counterpart

$$\min_{x \in \mathcal{X}} \sup_{\xi \in \mathcal{U}} z(x, \xi).$$

The considered uncertainty set often strongly influences the solvability and the solution approaches. A *finite uncertainty set* consists of finitely many scenarios, whereas, in an *interval uncertainty set*, the coefficients vary in intervals independently of each other. If the coefficients vary in intervals, but only a given number of coefficients may differ from their minimal values, we speak of *cardinality-constrained uncertainty* [BS03].

### 1.4 Multi-objective robust optimization

If several objective functions and uncertainties in (some of) these functions are given, we obtain a *multi-objective uncertain optimization problem*.

**Definition 6.** Given a feasible set of solutions  $\mathcal{X}$ , an uncertainty set  $\mathcal{U}$  and a multi-objective function  $z : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}^k$ , the family of multi-objective optimization problems

$$\left( \min_{x \in \mathcal{X}} z(x, \xi), \xi \in \mathcal{U} \right) \tag{1}$$

is called a multi-objective uncertain optimization problem.

There are several definitions of robust efficiency for multi-objective uncertain problems (see, e.g., [IS16]). The concept of minmax robust optimality for single-objective uncertain problems has been generalized to several objectives in various ways, since the notion of *worst case* is not clear in the multi-objective case. An intuitive concept, introduced by Kuroiwa and Lee [KL12], is to determine the worst case independently for each objective (see Definition 7). This yields a single vector for each solution and these vectors can be compared using the methods of multi-objective optimization.

**Definition 7.** A solution  $x \in \mathcal{X}$  is robust efficient for Problem (1) if  $x$  is an efficient solution for the robust counterpart

$$\min_{x \in \mathcal{X}} z^R(x) = \begin{pmatrix} \sup_{\xi \in \mathcal{U}} z_1(x, \xi) \\ \vdots \\ \sup_{\xi \in \mathcal{U}} z_k(x, \xi) \end{pmatrix}.$$

**Remark 8.** In this paper, we only consider uncertainty sets where the uncertainties in the objectives are independent of each other. That means that robust efficiency, as defined in Definition 7, is the same as point-based and set-based minmax robust efficiency defined in [EIS14]. Therefore, all results shown in this paper are valid for both concepts.

Analogously to Definition 4 we define:

**Definition 9.** Two robust efficient solutions  $x, x' \in \mathcal{X}$  are called equivalent if  $z^R(x) = z^R(x')$ . A set of robust efficient solutions  $\bar{\mathcal{X}} \subseteq \mathcal{X}$  is called complete if all  $x \in \mathcal{X} \setminus \bar{\mathcal{X}}$  are either dominated w.r.t.  $z^R$  or equivalent to at least one  $x' \in \bar{\mathcal{X}}$ .

## 1.5 Multi-objective robust combinatorial optimization

An instance  $(E, Q, \mathcal{U}, c)$  of a multi-objective uncertain combinatorial optimization problem is given by a finite element set  $E$ , a set  $Q \subseteq 2^{|E|}$  of feasible solutions, which are subsets of  $E$ , an uncertainty set  $\mathcal{U}$  and a function  $c$ , that assigns a *cost* vector  $c_e^\xi = (c_{e,1}^\xi, \dots, c_{e,k}^\xi)$  to each element  $e \in E$  and scenario  $\xi \in \mathcal{U}$ . For each scenario  $\xi$  the *cost*  $z(q, \xi)$  of a set  $q$  with respect to  $\xi$  is the sum of the costs of its elements. We aim to find a complete set of robust efficient solutions (according to Definition 7) for

$$\left( \min_{q \in Q} z(q, \xi) = \sum_{e \in q} c_e^\xi, \xi \in \mathcal{U} \right),$$

i.e., to find a complete set of efficient solutions for the robust counterpart

$$\min_{q \in Q} \begin{pmatrix} \max_{\xi \in \mathcal{U}} \sum_{e \in q} c_{e,1}^\xi \\ \vdots \\ \max_{\xi \in \mathcal{U}} \sum_{e \in q} c_{e,k}^\xi \end{pmatrix}.$$

## 1.6 Example: The 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 termination node  $t \in V$ . Let  $\mathcal{U}$  be an uncertainty set and  $c$  be a function that assigns a *cost* or *length*  $c_e^\xi = (c_{e,1}^\xi, \dots, c_{e,k}^\xi)$  to each edge  $e \in E$  and scenario  $\xi \in \mathcal{U}$ . For a path  $q$  in  $G$  and a scenario  $\xi \in \mathcal{U}$  the *cost* or *length*  $z(q, \xi)$  of  $q$  w.r.t.  $\xi$  is obtained by following the path and adding up the costs  $c_e^\xi$  of the edges traversed.

We distinguish between *simple paths*, which contain each node at most once and *paths*, which may contain nodes and edges more than once. In the deterministic case, there

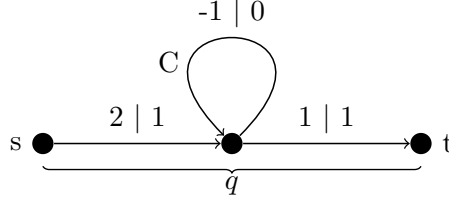


Figure 1: In Example 10 every robust shortest path contains a cycle.

always either exists a simple path being a shortest path, or no finite shortest path exists. On the contrary, robust shortest paths that contain a cycle but are not optimal without the cycle can exist, even in case of only one objective (see Example 10).

In the following we assume *conservative* edge costs, i.e., every cycle  $C$  has non-negative cost  $z(C, \xi) \geq 0$  for each scenario  $\xi \in \mathcal{U}$  and objective  $i = 1, \dots, k$ . Then, there always exists a complete set of robust efficient paths containing only simple paths and the *multi-objective uncertain shortest path problem* is

$$\left( \min_{q \in Q} z(q, \xi) = \sum_{e \in q} c_e^\xi, \xi \in \mathcal{U} \right)$$

with  $Q$  being the set of simple paths from  $s$  to  $t$  in  $G$ . Because simple paths do not contain any edge more than once, this is a combinatorial optimization problem.

The following single-objective example shows that, when edge costs are not conservative, we can indeed have robust shortest paths which contain cycles while no simple robust shortest path exists.

**Example 10.** Let  $G$  be a graph that consists of a simple path  $q$  from  $s$  to  $t$  and a cycle  $C$  connected to  $q$  (Figure 1). Let two scenarios  $\xi_1, \xi_2$  be given and let the cost of  $C$  be  $z(C, \xi_1) = -1$  and  $z(C, \xi_2) = 0$  and the cost of  $q$  be  $z(q, \xi_1) = 3$  and  $z(q, \xi_2) = 2$ . Let  $q^i$  for  $i \in \mathbb{N}$  denote the path that consists of  $q$  and  $i$  times the cycle  $C$ . Then,

$$\max_{\xi \in \{\xi_1, \xi_2\}} z(q, \xi) = 3 > 2 = \max_{\xi \in \{\xi_1, \xi_2\}} z(q^1, \xi) = \max_{\xi \in \{\xi_1, \xi_2\}} z(q^i, \xi) \quad \forall i \geq 1,$$

and  $q$  is not robust optimal, but  $q^1$  is robust optimal.

We use the following notation to specify subpaths.

**Notation 11.** 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  $w$ .

## 2 Algorithms for multi-objective combinatorial optimization problems with cardinality-constrained uncertainty

The idea of cardinality-constrained uncertainty is to assume that the worst case will not happen for all edges simultaneously, e.g., there will not be an accident on every road of a transportation network at the same time. Therefore, only those scenarios are considered where no more than a given number of elements are more expensive than their minimum costs. Bertsimas and Sim were the first to introduce cardinality-constrained uncertainty for single-objective uncertain optimization problems [BS03]. One possibility to extend this concept to multi-objective optimization is the following (see [HNS13]):

**Definition 12.** For each element  $e \in E$  and each objective  $z_i$  let two real values  $\hat{c}_{e,i}$  and  $\delta_{e,i} \geq 0$  be given. We assume that the uncertain cost  $c_{e,i}$  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  $z_i$  let an integer  $\Gamma_i \leq |E|$  be given. The cardinality-constrained uncertainty set contains all scenarios, in which for each scenario  $i$  at most  $\Gamma_i$  elements differ from their nominal costs:

$$\mathcal{U} := \{c \in \mathbb{R}^{|E| \times k} : c_{e,i} \in [\hat{c}_{e,i}, \hat{c}_{e,i} + \delta_{e,i}] \forall e \in E, \forall i = 1, \dots, k, |\{e : c_{e,i} > \hat{c}_{e,i}\}| \leq \Gamma_i \forall i = 1, \dots, k\}$$

An instance of a multi-objective combinatorial optimization problem with cardinality-constrained uncertainty is hence given by  $(E, Q, \hat{c}, \delta, \Gamma = (\Gamma_1, \dots, \Gamma_k))$ .

## 2.1 Deterministic Subproblems Algorithm (DSA)

The algorithms in this subsection are built upon an algorithm by Bertsimas and Sim for single-objective cardinality-constrained uncertain combinatorial optimization problems [BS03], 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.

We describe first the algorithm of Bertsimas and Sim for single-objective problems and present several ways to reduce the number of subproblems to be solved (Section 2.1.1). In Section 2.1.2, we show that DSA can be adjusted for multi-objective problems with a special property. Lastly, we extend the algorithm for the general multi-objective case in Section 2.1.3.

### 2.1.1 DSA for one objective

We first consider the single-objective uncertain problem  $(\min_{q \in Q} z(q, \xi), \xi \in \mathcal{U})$  with

$$\mathcal{U} = \{c \in \mathbb{R}^{|E|} : c_e \in [\hat{c}_e, \hat{c}_e + \delta_e] \forall e \in E, |\{e : c_e > \hat{c}_e\}| \leq \Gamma\}.$$

The worst case for a set  $q \in Q$  with respect to this uncertainty is a scenario, where the costs of its  $\Gamma$  elements with the largest intervals  $\delta_e$  take their maximal value (resp. all elements in  $q$ , if  $q$  has less than  $\Gamma$  elements). Assume that the elements are ordered with respect to the interval length  $\delta$ , i.e.,

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

For each  $l \in \{1, \dots, |E| + 1\}$  we define the function  $g^l$  as follows [BS03]:

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

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  $\delta_e$  has to be added to the costs for (at most)  $\Gamma$  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 of [BS03] is to solve all problems

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

for  $l = 0, 1, \dots, |E| + 1$  and choose the best of the obtained solutions. This idea works due to the following two properties:

1. For every set  $q$  and every  $l \in \{0, \dots, |E| + 1\}$  we have that  $g^l(q)$  is always greater than or equal to the worst case cost  $z^R(q)$ .
2. For every set  $q$  there exists some  $l \in \{0, \dots, |E| + 1\}$  such that  $g^l(q)$  equals the worst case cost  $z^R(q)$ .

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  $z^R(q) = \sum_{e \in q} \hat{c}_e + \sum_{j=1}^h \delta_{e_{a_j}}$  and we get

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

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

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

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

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

Having these two properties, we see that a robust optimal solution  $q^*$  is also optimal for problem  $(\mathcal{P}(l))$  with  $l : g^l(q) = z^R(q)$ , since none of the other sets can have a better objective value. Therefore, at least one robust optimal solution will be found by the iterative algorithm.

The efficiency of the algorithm 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

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

Algorithm 1 shows the basic structure of the algorithm by Bertsimas and Sim ([BS03]). First, the elements are ordered with respect to their interval lengths, then the subproblems defined above are solved and finally the worst case values of all obtained solutions are compared to find the robust optimal ones. Because the solutions of each subproblem can be obtained by solving a deterministic problem of the same kind, this algorithm finds a robust optimal solution in polynomial time for many combinatorial optimization problems, e.g., for the minimum spanning tree and the shortest path 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 from [BS03, PL07, LK14] can be used to reduce the number of subproblems (Lemma 13): First, if two elements have the same interval length  $\delta_e$ , then their associated subproblems are

---

**Algorithm 1** Basic structure of DSA (based on [BS03])

---

**Input:** an instance  $I = (E, Q, \hat{c}, \delta, \Gamma)$  of a single-objective cardinality-constrained uncertain combinatorial optimization problem

**Output:** a robust efficient solution for  $I$

- 1: Sort  $E$  w.r.t.  $\delta_e$  such that  $\bar{\delta}_1 := \delta_{e_1} \geq \bar{\delta}_2 := \delta_{e_2} \geq \dots \geq \bar{\delta}_{|E|} \geq \bar{\delta}_{|E|+1} := 0$ .
  - 2: Determine  $L := \{1, \dots, |E| + 1\}$ .
  - 3: For all  $l \in L$  find an optimal solution  $q^l$  for  $(\mathcal{P}(l))$ .
  - 4: Compare the objective values  $z^R(q^l)$  for all  $l \in L$  to obtain a robust optimal solution.
- 

equal. Second, the worst case cost of a set equals its objective value not only for the special subproblem shown above, but also for the next subproblem. 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  $\Gamma$ -th smallest index for any set in  $Q$ , so their associated subproblems need not to be solved.

**Lemma 13** ([BS03, PL07, LK14]). *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  $e_l, \dots, e_{(l+h)}$  with the same interval length  $\delta_{e_l} = \dots = \delta_{e_{l+h}}$ , only one of the subproblems  $\mathcal{P}(l), \dots, \mathcal{P}(l+h)$  needs to be solved [BS03].*
2. *Only every second subproblem and the last subproblem need to be solved [LK14].*
3. *It is sufficient to start with the  $\Gamma$ -th subproblem [PL07].*

Using these results we can replace Step 2 of the basic structure with Algorithm 2.

---

**Algorithm 2** Improved Step 2 of Algorithm 1: Determine the subproblems to be solved.

---

**Input:** an edge set  $E$  with cost interval lengths  $\delta$ , a value  $\Gamma \leq |E|$

**Output:** an index set  $L$  of subproblems to be solved in Algorithm 1

- 1:  $l := \Gamma + 1$   $\triangleright$  Lemma 13 (3., 2.)
  - 2:  $L := \{l\}$
  - 3: **while**  $l < |E| + 1$  **do**
  - 4:     **while**  $l < |E| + 1$  and  $\delta_{e_l} = \delta_{e_{l+1}}$  **do**  $l := l + 1$   $\triangleright$  Lemma 13 (1.)
  - 5:     **end while**
  - 6:     **if**  $l < |E| + 1$  **then**  $l := l + 1$
  - 7:         **if**  $l < |E| + 1$  **then**  $l := l + 1$   $\triangleright$  Lemma 13 (2.)
  - 8:         **end if**
  - 9:          $L := L \cup \{l\}$
  - 10:     **end if**
  - 11: **end while**
- 

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

**Lemma 14.** *Let  $1 \leq \tilde{l} \leq l \leq |E| + 1$  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  $e_1, \dots, e_{l-1}$ , 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 (\*). For these costs we have

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

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

$$\sum_{e \in q^{\tilde{l}}} c_e^l = \sum_{e \in q^{\tilde{l}}} c_e^{\tilde{l}} \leq \sum_{e \in q} c_e^{\tilde{l}} \leq \sum_{e \in q} c_e^l \quad \forall q \in Q,$$

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

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

---

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

---

**Input:**  $I = (E, Q, \hat{c}, \delta, \Gamma)$  with  $E$  ordered w.r.t.  $\delta_e$ ,  $\bar{\delta}$ , 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  $\{e_1, \dots, e_{l-1}\}$  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

```

---

Lemma 14 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 3 show the practical use of this improvement.

### 2.1.2 Extension to the multi-objective problem with objective-independent element order

In this section we adjust Algorithm 1 for multi-objective problems with the following property:

**Definition 15.** An instance  $(E, Q, \hat{c}, \delta, \Gamma)$  has objective independent element order if

- there exists an order of the elements, such that

$$\delta_{e_1, i} \geq \dots \geq \delta_{e_{|E|}, i} \quad \forall i = 1, \dots, k,$$

- and for all objective functions, the number of elements that may differ from the nominal value is the same, that is  $\Gamma_1 = \Gamma_2 = \dots = \Gamma_k$ . In the following we use  $\Gamma_1$  to denote the bound on the number of elements that may differ from the nominal value for each individual objective function.

For multi-objective subproblems with objective independent element order, Algorithm 1 can be adjusted in the following way:

In step 3, since  $\Gamma, \delta_e, \hat{c}_e$  are vectors instead of scalars, the subproblems to be solved are the multi-objective problems

$$(\mathcal{MP}(l)) \min_{q \in Q} g^l(q) := \sum_{e \in q} \hat{c}_e + \Gamma \circ \bar{\delta}_l + \sum_{\substack{e_j \in q \\ j \leq l}} (\delta_{e_j} - \bar{\delta}_l)$$

for  $l = 1, \dots, |E|+1$ , with  $\circ$  being the Schur (entry-wise) product and  $\bar{\delta}_{|E|+1} = (0, \dots, 0)$ ,  $\bar{\delta}_j := \delta_{e_j} \forall e_j \in E$ . Because we solve multi-objective problems, we are looking for a complete set of efficient solutions for each subproblem instead of a single solution. Such a solution set can be found by solving a deterministic multi-objective problem. We denote the solution set, that we obtain for  $(\mathcal{MP}(l))$ , by  $OPT^l$ .

In Step 4, every found solution  $q$  whose objective vector  $z^R(q)$  is not dominated by the objective vector of any of the other solutions is robust efficient. We will refer to this special version of the DSA for multi-objective instances with objective independent element order as *objective independent DSA* or *DSA-oi*.

---

**Algorithm 4** DSA for multi-objective instances with objective independent element order (DSA-oi)

---

**Input:** an instance  $I = (E, Q, \hat{c}, \delta, \Gamma)$  of a multi-objective cardinality-constrained uncertain combinatorial optimization problem with objective independent element order

**Output:** a complete set of robust efficient solutions for  $I$

- 1: Sort  $E$  w.r.t.  $\delta_e$  such that  $\bar{\delta}_1 := \delta_{e_1} \geq \bar{\delta}_2 := \delta_{e_2} \geq \dots \geq \bar{\delta}_{|E|} \geq \bar{\delta}_{|E|+1} := (0, \dots, 0)$ .
  - 2: Determine  $L := \{1, \dots, |E| + 1\}$ .
  - 3: For all  $l \in L$  find a complete set of efficient solutions  $OPT^l$  for  $(\mathcal{MP}(l))$ .
  - 4: Compare the objective vectors  $z^R(q)$  of all solutions in  $\cup_{l \in L} OPT^l$ . The solutions with non-dominated objective vectors form a complete set of robust efficient solutions.
- 

**Theorem 16.** *Algorithm 4 finds a complete set of robust efficient solutions for multi-objective cardinality-constrained uncertain combinatorial optimization problems with objective independent element order.*

*Proof.* First, we show that  $g^l$  never underestimates  $z^R$  for any objective. Further, we prove that for each feasible solution  $q$  there is an  $l \in \{\Gamma_1, \dots, |E| + 1\} \subseteq L$  with  $g^l(q) = z^R(q)$ . We conclude that Algorithm 4 finds a complete set of robust efficient solutions.

For each  $q \in Q$  and  $l = 1, \dots, |E|+1$  we show  $z_i^R(q) \leq g_i^l(q) \forall i = 1, \dots, k$ . Let  $\{e_{a_1}, \dots, e_{a_h}\}$  be a set of  $h$  elements in  $q$  with the largest cost intervals, where  $h = \min\{|q|, \Gamma_i\}$ . Then,

$$\begin{aligned} g_i^l(q) &= \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}_{l,i} + \sum_{\substack{e_j \in q \\ j \leq l}} (\delta_{e_j,i} - \bar{\delta}_{l,i}) \\ &= \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}_{l,i} + \sum_{e \in q} \max\{0, \delta_{e,i} - \bar{\delta}_{l,i}\} && \text{since } j \leq l \Rightarrow \delta_{e_j} \geq \bar{\delta}_l, j > l \Rightarrow \delta_{e_j} \leq \bar{\delta}_l \\ &\geq \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}_{l,i} + \sum_{j=1}^h \max\{0, \delta_{e_{a_j},i} - \bar{\delta}_{l,i}\} && \text{since } \{e_{a_1}, \dots, e_{a_h}\} \subseteq q \\ &\geq \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}_{l,i} + \sum_{j=1}^h (\delta_{e_{a_j},i} - \bar{\delta}_{l,i}) \\ &\geq \sum_{e \in q} \hat{c}_{e,i} + \sum_{j=1}^h \bar{\delta}_{l,i} + \sum_{j=1}^h (\delta_{e_{a_j},i} - \bar{\delta}_{l,i}) = z_i^R(q) && \text{since } |\{e_{a_1}, \dots, e_{a_h}\}| \leq \Gamma_i. \end{aligned}$$

We show now that there is an  $l \in \{\Gamma_1, \dots, |E| + 1\}$  with  $g^l(q) = z^R(q)$ : For any set  $q \in Q$  with at least  $\Gamma_1$  elements, let  $e_l$  be the element with the  $\Gamma_1$ -th smallest index. Then the  $\Gamma_1$  elements  $\{e_j \in q : j \leq l\}$  have the largest cost intervals in  $q$  with respect to every objective. It follows for all  $i = 1, \dots, k$  that

$$\begin{aligned} g_i^l(q) &= \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot \bar{\delta}_{l,i} + \sum_{\substack{e_j \in q \\ j \leq l}} (\delta_{e_j,i} - \bar{\delta}_{l,i}) \\ &= \sum_{e \in q} \hat{c}_{e,i} + \sum_{\substack{e_j \in q \\ j \leq l}} \bar{\delta}_{l,i} + \sum_{\substack{e_j \in q \\ j \leq l}} (\delta_{e_j,i} - \bar{\delta}_{l,i}) = z_i^R(q) \quad \text{since } |\{e_j \in q : j \leq l\}| = \Gamma_i. \end{aligned}$$

For any set  $q \in Q$  with less than  $\Gamma_1$  elements, we have for all  $i = 1, \dots, k$

$$g_i^{|E|+1}(q) = \sum_{e \in q} \hat{c}_{e,i} + \Gamma_i \cdot 0 + \sum_{e_j \in q} (\delta_{e_j,i} - 0) = z_i^R(q).$$

We conclude: If  $q$  is robust efficient, then  $z^R(q) = g^l(q)$  for some  $l \in L$  and there is no  $q' \in Q$  with  $z^R(q') \leq z^R(q)$ . It follows that

$$\nexists q' \in Q : z^R(q') \leq z^R(q) \xrightarrow{z^R(q') \leq g^l(q')} \nexists q' \in Q : g^l(q') \leq z^R(q) = g^l(q).$$

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.  $\square$

Now, we consider the enhancements proposed in Algorithms 2 and 3. The results of Lemma 13 remain valid, Step 2 can hence be implemented as in Algorithm 2.

**Lemma 17.** *The number of subproblems to be solved by Algorithm 4 can be reduced to  $\left\lceil \frac{|E| - \Gamma_1}{2} \right\rceil + 1$  in the same ways as in the single-objective case:*

1. *If there are several elements  $e_l, \dots, e_{l+h}$  with the same interval length  $\delta_{e_l} = \dots = \delta_{e_{l+h}}$ , only one of the subproblems  $\mathcal{MP}(l), \dots, \mathcal{MP}(l+h)$  needs to be solved.*
2. *Only every second subproblem and  $\mathcal{MP}(|E| + 1)$  need to be solved.*
3. *It is sufficient to start with  $\mathcal{MP}(\Gamma_1)$ .*

*Proof.*

1. From  $\delta_{e_l} = \dots = \delta_{e_{l+h}}$  follows directly  $g^l(q) = \dots = g^{l+h}(q)$  and therefore  $OPT^l(q) = \dots = OPT^{l+h}(q)$ .
2. For any  $q \in Q$  with less than  $\Gamma_1$  elements we have  $z^R(q) = g^{|E|+1}(q)$ . For any  $q \in Q$  with at least  $\Gamma_1$  elements let  $e_l$  be the element with the  $\Gamma_1$ -th smallest index in  $q$ . From the proof of Theorem 16 we know that  $z^R(q) = g^l(q)$ . We further have

$$\begin{aligned} g^l(q) &= \sum_{e \in q} \hat{c}_e + \Gamma \circ \bar{\delta}_l + \sum_{\substack{e_j \in q \\ j \leq l}} (\delta_{e_j} - \bar{\delta}_l) + \Gamma \circ (\bar{\delta}_{l+1} - \bar{\delta}_l) + \Gamma \circ (\bar{\delta}_l - \bar{\delta}_{l+1}) \\ &= \sum_{e \in q} \hat{c}_e + \Gamma \circ \bar{\delta}_{l+1} + \sum_{\substack{e_j \in q \\ j \leq l}} (\delta_{e_j} - \bar{\delta}_{l+1}) \quad \text{because } |\{e_j \in q : j \leq l\}| = \Gamma_i \quad \forall i \\ &= \sum_{e \in q} \hat{c}_e + \Gamma \circ \bar{\delta}_{l+1} + \sum_{\substack{e_j \in q \\ j \leq l+1}} (\delta_{e_j} - \bar{\delta}_{l+1}) = g^{l+1}(q). \end{aligned}$$

Therefore, if we only solve every second subproblem, we will still solve at least one subproblem  $\mathcal{MP}(l')$  with  $g^{l'}(q) = z^R(q)$  for each  $q \in Q$  with at least  $\Gamma_1$  elements. It follows, that we only need to solve every second subproblem and in addition the  $(|E| + 1)$ -th subproblem.

3. In the proof of Theorem 16 we show that for every  $q \in Q$  there is an  $l \in \{\Gamma_1, \dots, |E| + 1\}$  with  $z^R(q) = g^l(q)$ , because none of the elements  $e_1, \dots, e_{\Gamma_1-1}$  can be the element with the  $\Gamma_1$ -th smallest index in  $q$ . It follows, that we do not need to solve the problems  $\mathcal{MP}(1), \dots, \mathcal{MP}(\Gamma_1 - 1)$  to find a complete set of robust efficient solutions.

From statement 2 we know that at most  $|E| + 1 - (\Gamma_1 - 1)$  problems need to be solved. From statement 3 it follows that of these problems only the last one and every second of the other ones must be solved, this leads to at most

$$\left\lfloor \frac{|E| + 1 - (\Gamma_1 - 1) - 1}{2} \right\rfloor + 1 = \left\lfloor \frac{|E| - \Gamma_1 + 1}{2} \right\rfloor + 1 = \left\lceil \frac{|E| - \Gamma_1}{2} \right\rceil + 1$$

subproblems.  $\square$

The result of Lemma 14 is valid for multi-objective problems with objective independent element order as well. However, to be able to skip the solving of problem  $\mathcal{MP}(l)$  none of the sets in  $OPT^{\tilde{l}}$  is allowed to contain any element  $e_j$  with  $j < l$ . Therefore we replace Step 3 with Algorithm 5.

---

**Algorithm 5** Improved step 3 of Algorithm 4: Solve subproblems (with solution checking).

---

**Input:**  $I = (E, Q, \hat{c}, \delta, \Gamma)$  with  $E$  ordered w.r.t.  $\delta_i, \bar{\delta}$ , an index set of subproblems  $L$

**Output:** a set of solutions  $\cup_{l \in L} OPT^l$

```

1:  $\tilde{l} := 0$ 
2: for all  $l \in L$  in increasing order do
3:   if  $\tilde{l} = 0$  or any of the sets in  $OPT^{\tilde{l}}$  contains any element in  $\{e_1, \dots, e_{l-1}\}$  then
4:     Find a complete set of efficient solutions  $OPT^l$  for  $(\mathcal{MP}(l))$ 
5:   else  $OPT^{\tilde{l}} := OPT^l$ 
6:   end if
7:    $\tilde{l} := l$ 
8: end for
```

---

**Lemma 18.** Let  $1 \leq \tilde{l} \leq l \leq |E| + 1$  and let  $G^{\tilde{l}}$  be a complete set of efficient solutions for  $\mathcal{MP}(\tilde{l})$ . If none of the sets in  $G^{\tilde{l}}$  contains any of the elements  $e_1, \dots, e_{l-1}$ , then  $G^{\tilde{l}}$  is a complete set of efficient solutions for  $\mathcal{MP}(l)$ .

*Proof.* A complete set of solutions for  $\mathcal{MP}(l)$  can be found by solving a deterministic multi-objective problem with costs  $c_e^l := (c_{e,1}^l, \dots, c_{e,k}^l)$ :

$$c_{e_j,i}^l := \begin{cases} \hat{c}_{e_j,i} + (\delta_{e_j,i} - \bar{\delta}_{l,i}) & \text{for } j < l \\ \hat{c}_{e_j,i} & \text{for } j \geq l. \end{cases}$$

From the proof of Lemma 13 we know that  $c_{e_j,i}^{\tilde{l}} \leq c_{e_j,i}^l \forall e_j : j < l$  and  $c_{e_j,i}^{\tilde{l}} = c_{e_j,i}^l \forall e_j : j \geq l$ . It follows that any  $q \in OPT^{\tilde{l}}$  not containing any element in  $\{e_1, \dots, e_{l-1}\}$  is also efficient w.r.t.  $c^l$ . If none of the sets in  $OPT^{\tilde{l}}$  contains any element in  $\{e_1, \dots, e_{l-1}\}$ , then for any  $q' \notin OPT^{\tilde{l}}$  exists a  $q \in OPT^{\tilde{l}}$  with

$$\sum_{e \in q} c_e^l = \sum_{e \in q} c_e^{\tilde{l}} \leq \sum_{e \in q'} c_e^{\tilde{l}} \leq \sum_{e \in q'} c_e^l$$

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

**Corollary 19.** *If we replace in Algorithm 4 Step 2 with Algorithm 2 and Step 3 with Algorithm 5, it finds a complete set of robust efficient solutions for multi-objective cardinality-constrained uncertain combinatorial optimization problems with objective independent element order, solving at most  $\left\lceil \frac{|E| - \Gamma_1}{2} \right\rceil + 1$  deterministic subproblems.*

### 2.1.3 The Deterministic Subproblems Algorithm in the general multi-objective case

In general, the sorting of the elements by interval lengths results in a different order for each objective. An element that has the  $\Gamma$ -th longest interval in  $q$  for all objectives 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.

Let  $E_j^i$  be a set of the  $j$  elements with the largest intervals for the  $i$ -th objective, i.e.,  $|E_j^i| = j$  and  $\delta_{e,i} \geq \delta_{e',i} \forall e \in E_j^i, e' \in E \setminus E_j^i$  and let  $\bar{\delta}_j^i := \min_{e \in E_j^i} \delta_{e,i}$ .

We define  $\bar{\delta}_{|E|+1}^i := 0 \forall i$ . For each  $l = (l_1, \dots, l_k) \in \{1, \dots, |E| + 1\} \times \dots \times \{1, \dots, |E| + 1\}$  we define the problem

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

As before, each of these  $(|E|+1)^k$  problems can be solved as a deterministic multi-objective problem of the same kind.

Algorithm 6 preserves the basic structure of DSA: First, the elements are sorted w.r.t.  $\delta_{e,i}$  for each  $i = 1, \dots, k$ . Instead of changing the indices, we store the set  $E_j^i$  of the first  $j$  elements for all  $j = 1, \dots, |E|$ , because the order of the elements depends on the objective. Then the 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.

---

#### Algorithm 6 DSA for general multi-objective instances

---

**Input:** an instance  $I = (E, Q, \hat{c}, \delta, \Gamma)$  of a multi-objective cardinality-constrained uncertain combinatorial optimization problem

**Output:** a complete set of robust efficient solutions for  $I$

- 1: For  $i := 1, \dots, k$ : Sort  $E$  w.r.t.  $\delta_{e,i}$  descending and save the first  $j$  elements in  $E_j^i$  for  $j = 1, \dots, |E|$ . Set  $E_{|E|+1}^i := E$ . Set  $\bar{\delta}_j^i := \min_{e \in E_j^i} \delta_{e,i} \forall j = 1, \dots, |E|$  and  $\bar{\delta}_{|E|+1}^i := 0$ .
  - 2: Determine  $L = L_1 \times L_2 \times \dots \times L_k$ :  $L_i := \{1, \dots, |E| + 1\} \forall i = 1, \dots, k$ .
  - 3: For all  $l \in L$  find a complete set of efficient solutions  $OPT^l$  for  $(\mathcal{GMP}(l))$ .
  - 4: Compare the objective vectors  $z^R(q)$  of all solutions in  $\cup_{l \in L} OPT^l$ . The solutions with non-dominated objective vectors form a complete set of robust efficient solutions.
- 

**Theorem 20.** *Algorithm 6 finds a complete set of robust efficient solutions for multi-objective cardinality-constrained uncertain combinatorial optimization problems.*

*Proof.* Analogously to the proof of Theorem 16 it can be shown that  $z^R(q) \leq g^l(q) \forall q \in Q$  and that for every  $q \in Q$  and  $i \in \{1, \dots, k\}$  there exists an  $l$  with  $l_i \in \{1, \dots, |E| + 1\}$  and  $z_i^R(q) = g_i^l(q)$ . Since we consider every combination of the values of  $l_1, \dots, l_k$  it follows that

for every  $q \in Q$  it exists a problem  $\mathcal{GMP}(l)$  with  $z_i^R(q) = g_i^l(q) \forall i = 1, \dots, k$ . It follows that a complete set of robust efficient solutions is returned.  $\square$

As before, we can reduce the number of subproblems to be solved. The proof of Lemma 17 still holds for each objective independently. Therefore, we can apply the reduction on each objective (Algorithm 7) and obtain  $L := L_1 \times L_2 \times \dots \times L_k$  with  $|L_i| = \left\lceil \frac{|E| - \Gamma_i}{2} \right\rceil + 1$ .

---

**Algorithm 7** Improved Step 2 of Algorithm 6: Determine the subproblems to be solved.

---

**Input:** an edge set  $E$  with cost interval lengths  $\delta$ , a  $k$ -dimensional vector  $\Gamma$  with  $\Gamma_i \leq |E| \forall i$

**Output:** an index set  $L$  of subproblems to be solved in Algorithm 6

- 1: **for**  $i = 1, \dots, k$  **do**
  - 2:     Determine  $L_i$  as in Algorithm 2 with  $\Gamma = \Gamma_i, \delta_e = \delta_{e,i}$ .
  - 3: **end for**
- 

Here again, we can use solution checking, i.e., skip some additional subproblems, depending on the solutions found so far.

**Lemma 21.** *Let  $l, \tilde{l} \in \mathbb{Z}^k$  be given with  $l \leq \tilde{l}$  and let  $I$  be the set of indices  $i$  with  $l_i < \tilde{l}_i$ . Let  $OPT^{\tilde{l}}$  be a complete set of efficient solutions for  $\mathcal{GMP}(\tilde{l})$ . If for all  $i$  for which  $l_i < \tilde{l}_i$ , none of the sets in  $OPT^{\tilde{l}}$  contains an element in  $\cup_{i \in I} E_{l_i}^i$ , then  $OPT^{\tilde{l}}$  is a complete set of efficient solutions for  $\mathcal{GMP}(l)$ .*

*Proof.* Since  $\Gamma_i \cdot \bar{\delta}_i^{l_i}$  are solution independent constants, the minimization problem to be solved is a deterministic multi-objective problem with costs  $c_e^l = (c_{e,1}^l, \dots, c_{e,k}^l)$ :

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

Therefore,

$$\begin{aligned} c_{e,i}^l &= c_{e,i}^{\tilde{l}} \quad \forall i \text{ with } l_i = \tilde{l}_i, \quad \forall e \in E \\ c_{e,i}^l &= c_{e,i}^{\tilde{l}} \quad \forall i \text{ with } l_i < \tilde{l}_i, \quad \forall e \in E \setminus E_{l_i}^i \\ c_{e,i}^{\tilde{l}} &\leq c_{e,i}^l \quad \forall i, \quad \forall e \in E. \end{aligned}$$

Hence, for all objective functions  $i$  we have  $c_{e,i}^l = c_{e,i}^{\tilde{l}}$  for all elements that are contained in any set in  $OPT^{\tilde{l}}$ , and  $c_{e,i}^{\tilde{l}} \leq c_{e,i}^l$  for all elements that are not contained in a set in  $OPT^{\tilde{l}}$ . Analogously to the proof of Lemma 18, it follows that  $OPT^{\tilde{l}}$  is a complete set of efficient solutions for  $\mathcal{GMP}(l)$  if no solution in  $OPT^{\tilde{l}}$  contains any element in  $\cup_{i \in I} E_{l_i}^i$ .  $\square$

A fast way to use this result is to replace Step 3 of Algorithm 6 with Algorithm 8.

**Lemma 22.** *In Line 8 of Algorithm 8, if  $\tilde{l}^h \neq (0, \dots, 0)$ , then  $1 \leq \tilde{l}_h < l_h \leq |E| + 1$  and  $\tilde{l}_i = l_i \forall i = 1, \dots, k, i \neq h$ .*

*Proof.* For every  $i = 1, \dots, k$  let  $l_i^1 := \min_{l_i \in L_i} l_i$  be the minimal element in  $L_i$ . We use the following observations:

1. Because  $h = 1$  and  $\tilde{l}^1 = (0, \dots, 0)$  in the first iteration, the first subproblem is solved and then  $\tilde{l}^i$  is set to  $\tilde{l}^i := (l_1^1, l_2^1, \dots, l_k^1)$  for all  $i = 1, \dots, k$  in Line 13. Thereafter,  $\tilde{l}^i$  is changed in Line 13 if and only if  $h \leq i$ .

---

**Algorithm 8** Improved step 3 of Algorithm 6: Solve subproblems (with solution checking).

---

**Input:** an instance  $I = (E, Q, \hat{c}, \delta, \Gamma)$ ,  $\bar{\delta}$ , edge sets  $E_j^i \forall i, j$ , an index set  $L$  of subproblems

**Output:** a set of solutions  $\cup_{l \in L} OPT^l$

```

1:  $\tilde{l}^1 := (0, \dots, 0)$ 
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:        $l := (l_1, \dots, l_k)$ 
8:       if  $\tilde{l}^h = (0, \dots, 0)$  or any of the sets in  $OPT^{\tilde{l}^h}$  contains any element in  $E_{l_h}^h$  then
9:         Find a complete set of efficient solutions  $OPT^l$  for  $(\mathcal{GMP}(l))$ .
10:      else  $OPT^l := OPT^{\tilde{l}^h}$ 
11:      end if
12:      for  $i = h, \dots, k$  do
13:         $\tilde{l}^i := l$ 
14:      end for
15:       $h := k$ 
16:    end for
17:    ...
18:     $h := 2$ 
19:  end for
20:   $h := 1$ 
21: end for

```

---

2. The value of  $h$  is changed to  $\hat{h}$  whenever one iteration of the for-loop changing  $l_{\hat{h}}$  is finished. Then  $l_i = l_i^1 \forall i > \hat{h}$  during the next execution of Lines 8 to 15.
3. In Line 8

$$h = \min\{i \in \{1, \dots, k\} : l_i \text{ has changed since the last execution of Line 8}\}.$$

We consider the state of the algorithm during any iteration of Line 8 and show  $\tilde{l}_{\hat{h}}^h < l_{\hat{h}}$  and  $\tilde{l}_i^h = l_i \forall i \neq \hat{h}$ . Let  $\hat{h}$  denote the value of  $h$  at this moment.

- $i > \hat{h}$ : When  $\tilde{l}^{\hat{h}}$  was changed last,  $h \leq \hat{h} < i$  hold (1.), so  $\tilde{l}_i^{\hat{h}}$  was set to  $l_i^1$  (2.). It follows  $\tilde{l}_i^{\hat{h}} = l_i^1 \stackrel{2.}{=} l_i$ .
- $i < \hat{h}$ : When  $\tilde{l}^{\hat{h}}$  was changed, either  $h < i$  or  $h \geq i$  hold. If it was  $h < i$  then  $\tilde{l}_i^{\hat{h}}$  was set to  $l_i^1$  (2.) and it follows  $\tilde{l}_i^{\hat{h}} = l_i^1 \stackrel{2.}{=} l_i$ .  
If it was  $h \geq i$ , then  $l_i$  was not changed since then, otherwise  $\tilde{l}^{\hat{h}}$  would have been changed again, because of  $i < \hat{h}$  (1.). It follows  $l_i = \tilde{l}_i^{\hat{h}}$ .
- $i = \hat{h}$ : We show first that  $l_{\hat{h}}$  changed at most once since the last change of  $\tilde{l}^{\hat{h}}$ . During the first execution of Line 8 after the first change of  $l_{\hat{h}}$  it holds  $h \geq \hat{h}$  (3.). So  $\tilde{l}^{\hat{h}}$  is changed again in Line 13, before  $l_{\hat{h}}$  could be changed a second time.  
From (3.) follows, that  $l_{\hat{h}}$  has changed since the last execution of Line 8, but  $l_{\hat{h}-1}$  hasn't. Therefore, the for-loop changing  $l_{\hat{h}-1}$  has not been finished. Hence,  $l_{\hat{h}}$  can not have been set to  $l_{\hat{h}}^1$  again, but was increased. This was the only change of  $l_{\hat{h}}$  since  $\tilde{l}^{\hat{h}}$  was set to its current value. It follows  $\tilde{l}_{\hat{h}}^{\hat{h}} < l_{\hat{h}}$ .

□

**Corollary 23.** *If we replace in Algorithm 6 Step 2 with Algorithm 2 and Step 3 with Algorithm 8, it still finds a complete set of robust efficient solutions for general instances of multi-objective cardinality-constrained uncertain combinatorial optimization problems. 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.*

The number of subproblems to be solved for general instances is hence a lot higher than for instances with objective independent element order. But if there exists a common element order for only a subset of the objectives, we can already reduce the number of subproblems significantly:

**Definition 24.** *An instance  $(E, Q, \hat{c}, \delta, \Gamma)$  has partial objective independent element order if there is a subset  $\{i_1, \dots, i_r\} \subset \{1, \dots, k\}$  with*

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

$$\delta_{e_1, i} \geq \dots \geq \delta_{e_{|E|}, i} \quad \forall i \in \{i_1, \dots, i_r\}.$$

**Lemma 25.** *Let an instance  $(E, Q, \hat{c}, \delta, \Gamma)$  with partial objective independent element order be given and let  $\{i_1, \dots, i_r\}$  be the subset from definition 24. Then the nested for-loops changing  $l_{i_1}, \dots, l_{i_r}$  in Algorithm 8 can be replaced by a single for-loop. The number of solved deterministic subproblems in Algorithm 6 with Algorithm 2 as Step 2 and Algorithm 8 as Step 3 is then less or equal to*

$$\left( \left\lceil \frac{|E| - \Gamma_{i_1}}{2} \right\rceil + 1 \right) \cdot \prod_{i \in \{1, \dots, k\} \setminus \{i_1, \dots, i_r\}} \left( \left\lceil \frac{|E| - \Gamma_i}{2} \right\rceil + 1 \right).$$

*Proof.* This follows directly from the proofs of Theorem 16 and Lemma 17.  $\square$

## 2.2 Bottleneck approach

DSA is especially useful for problems with high  $\Gamma_i$ , because fewer subproblems have to be solved for higher values of  $\Gamma_i$ . For small  $\Gamma_i$  the method described in 2.2.1 might be preferred. Its idea is to transfer the multi-objective uncertain combinatorial optimization problem 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 2.2.2.

### 2.2.1 Bottleneck approach for cardinality-constrained uncertain combinatorial optimization problems

We first consider the single-objective uncertain problem  $(\min_{q \in Q} z(q, \xi), \xi \in \mathcal{U})$ . Its min-max robust counterpart is

$$\min_{q \in Q} \left( z^R(q) := \max_{\xi \in \mathcal{U}} z(q, \xi) \right). \quad (2)$$

**Definition 26.** *For a set  $A \subseteq \mathbb{R}$  let  $j\text{-max}(A)$  denote the  $j$ -greatest element of  $A$ . For a set  $q \subseteq E$  let  $j\text{-max}_{e \in q} \delta_e := j\text{-max}(\{\delta_e : e \in q\})$  denote the  $j$ -highest value  $\delta_e$  which appears in  $q$ . If  $q$  has less than  $j$  elements we define  $j\text{-max}_{e \in q} \delta_e := 0$*



**Theorem 27.** *Every optimal solution for (2) is an efficient solution for the deterministic multi-objective problem*

$$\min_{q \in Q} z^L(q) := \begin{pmatrix} \sum_{e \in q} \hat{c}_e \\ \max_{e \in q} \delta_e \\ 2\text{-}\max_{e \in q} \delta_e \\ \vdots \\ \Gamma\text{-}\max_{e \in q} \delta_e \end{pmatrix}. \quad (3)$$

*Proof.* Let  $q$  be an optimal solution for Problem (2). Assume that  $q$  is not efficient for Problem (3). Then there exists a solution  $q' \in Q$  that dominates  $q$  and it follows

$$\begin{aligned} \sum_{e \in q'} \hat{c}_e &\leq \sum_{e \in q} \hat{c}_e \text{ and } j\text{-}\max_{e \in q'} \delta_e \leq j\text{-}\max_{e \in q} \delta_e \quad \forall j = 1, \dots, \Gamma, \text{ with at least one inequality} \\ \Rightarrow z^R(q') &= \sum_{e \in q'} \hat{c}_e + \sum_{j=1}^{\Gamma} j\text{-}\max_{e \in q'} \delta_e < \sum_{e \in q} \hat{c}_e + \sum_{j=1}^{\Gamma} j\text{-}\max_{e \in q} \delta_e = z^R(q), \end{aligned}$$

because the worst case scenario for any feasible set is a scenario where the cost of its  $\Gamma$  elements with the largest cost intervals take their maximal values. This contradicts  $q$  being optimal for (2).  $\square$

The reverse of Theorem 27 does not hold: There exist efficient solutions for (3), which are not optimal for (2), as the following example shows.

**Example 28.** *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 Problem (3), because*

$$z^L(q) = (3, 0, 0) \not\leq (0, 1, 1) = z^L(q') \text{ and } z^L(q') = (0, 1, 1) \not\leq (3, 0, 0) = z^L(q).$$

*But only  $q'$  is robust efficient, because*

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

**Lemma 29.** *A complete set of efficient solutions for Problem (3) contains at least one optimal solution for Problem (2).*

*Proof.* Let  $Q' \subseteq Q$  be a complete set of efficient solutions for (3). Assume, that (2) has an optimal solution  $q$  that is not contained in  $Q'$ . According to Lemma 27,  $q$  is an efficient solution for Problem (3), so  $Q'$  contains a solution  $q'$  with

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

and  $q'$  is optimal for (2).  $\square$

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

**Theorem 30.** *Every efficient solution for the multi-objective robust counterpart*

$$\min_{q \in Q} z^R(q) = \begin{pmatrix} \max_{\xi \in \mathcal{U}} z_1(q, \xi) \\ \vdots \\ \max_{\xi \in \mathcal{U}} z_k(q, \xi) \end{pmatrix} \quad (4)$$

*is an efficient solution for the deterministic multi-objective problem*

$$\min_{q \in Q} z^L(q) := \begin{pmatrix} \sum_{e \in q} \hat{c}_{e,1} \\ \max_{e \in q} \delta_{e,1} \\ 2\text{-}\max_{e \in q} \delta_{e,1} \\ \vdots \\ \Gamma_1\text{-}\max_{e \in q} \delta_{e,1} \\ \sum_{e \in q} \hat{c}_{e,2} \\ \max_{e \in q} \delta_{e,2} \\ \vdots \\ \Gamma_k\text{-}\max_{e \in q} \delta_{e,k} \end{pmatrix}. \quad (5)$$

*A complete set of solutions for (5) contains a complete set of solutions for (4).*

*Proof.* Let  $q$  be an efficient solution for Problem (4). Assume that  $q$  is not efficient for Problem (5). Analogously to the proof of Lemma 27, there is a path  $q' \in Q$  dominating  $q$  and it follows that  $z_i^R(q') < z_i^R(q)$  for at least one  $i \in \{1, \dots, k\}$ , which contradicts  $q$  being efficient for (4).

Assume now, that  $q \notin Q'$  with  $Q'$  being a complete set of efficient solutions for Problem (5). Since  $q$  is efficient for Problem (5), there is a solution  $q' \in Q'$  equivalent to  $q$  w.r.t. the objective function of Problem (5) and it follows  $z^R(q) = z^R(q')$  analogously to the proof of Lemma 29.  $\square$

With an algorithm solving the deterministic Problem (5) 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, an algorithm to solve Problem (3) was introduced in [GKR12].

### 2.2.2 Label setting algorithm (LSA) for the cardinality-constrained uncertain shortest path problem

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 the cardinality-constrained uncertain shortest path problem be defined as in Section 1.6, 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  $t$ . For simplicity we consider the single-objective uncertain shortest path problem, i.e., we show how to solve Problem (3), but the adjustments can be used for Problem (5) in the same way. Additionally we assume non-negative edge costs ( $\hat{c}_e \geq 0 \forall e \in E$ ) 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  $l = (z, v', l')$  at a node  $v$  consists of

- a cost vector  $z$ , here  $z = (z_0, \dots, z_\Gamma)$ ,
- 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 9 is a label setting algorithm for solving a shortest path problem of type (3). It is based on the label setting algorithm of Martins for multi-objective shortest path problems [Mar84], but we make the following adjustments:

1. In Step 3 a label must be chosen whose cost is not dominated by the cost of any other temporary label. In [Mar84] the lexicographically smallest label is chosen. Based on [IMP10], we choose the label with the smallest aggregate function  $\sum_{i=0, \dots, \Gamma} z_i$  instead.
2. In multi-objective label setting algorithms with only sum functions (as in [Mar84]) a new label  $l = (z, v', l')$  at  $v$  is created by adding the cost  $z'$  of 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 [GBR06]). To solve Problem (3) we need a new way to construct the labels: For the sum objective function, we add the nominal costs  $\hat{c}_e$  of the edge  $e := (v', v)$  to the corresponding predecessor cost entry  $z_0$ . For the  $j$ -max objective functions, we compare the interval length  $\delta_e$  of  $e$  to each of the  $\Gamma$  longest interval lengths so far  $z'_1, \dots, z'_\Gamma$  and insert it at the right position (see Algorithm 10). We will use the following notation:  $z := z' \oplus (\hat{c}_e, \delta_e)$ .
3. In [Mar84] 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 were introduced by [GBR06] for problems with bottleneck functions. Since new labels with the same cost as existing labels are immediately deleted, Algorithm 9 works even without the assumption that no cycles of cost  $(0, \dots, 0)$  exist.

---

**Algorithm 9** Label setting algorithm to solve a shortest path problem of type (3)

---

**Input:** an instance  $I = (E, Q, \hat{c}, \delta, \Gamma)$  of a multi-objective shortest path problem of type (3)

**Output:** permanent labels at  $t$ , representing a complete set of efficient solutions for  $I$

---

- 1: Create a temporary label  $l_0$  with cost  $(0, \dots, 0)$  at node  $s$ .
  - 2: **while** there exists at least one temporary label **do**
  - 3:     Select a temporary label  $l'$  (at any node  $v'$ ) with minimal aggregate cost  $\sum_{i=0, \dots, \Gamma} z'_i$  and make it permanent.
  - 4:     **for all** outgoing edges  $(v', v)$  of  $v'$  **do**
  - 5:         Create a new temporary label  $l$  at  $v$  by Algorithm 10.
  - 6:         **if** the cost of  $l$  is dominated by or equal to the cost of another label at  $v$  **then**
  - 7:             Delete  $l$ .
  - 8:         **else if**  $l$  dominates any temporary labels at  $v$  **then**
  - 9:             Delete these labels.
  - 10:        **end if**
  - 11:     **end for**
  - 12: **end while**
- 

**Lemma 31.** *In Algorithm 9 for every label  $l = (z, v', l')$  at a node  $v$  there exists a path  $q$  from  $s$  to  $v$  with  $z = z^L(q)$ .*

---

**Algorithm 10** Step 5 of Algorithm 9: Create a new temporary label.

---

**Input:** an edge  $(v', v)$ , a label  $l'$  with cost  $z'$  at a node  $v'$

**Output:** a new label  $l$  at  $v$  with predecessor label  $l'$

```

1:  $z_0 := z'_0 + \hat{c}_{(v', v)}$ 
2:  $i := 1$ 
3: while  $i \leq \Gamma$  do
4:   if  $\delta_{(v', v)} > z'_i$  then
5:      $z_i := \delta_{(v', v)}$ 
6:     for  $j := i + 1, \dots, \Gamma$  do  $z_j := z'_{j-1}$ 
7:   end for
8:    $i := \Gamma + 1$ 
9: else
10:   $z_i := z'_i$ 
11:   $i := i + 1$ 
12: end if
13: end while
14: Create the temporary label  $l := ((z_0, \dots, z_\Gamma), v', l')$  at node  $v$ .
```

---

*Proof.* We show the statement by induction:

The first label has cost  $(0, \dots, 0)$  and represents the path only consisting of node  $s$ .

Let  $z' = (z'_0, \dots, z'_\Gamma)$  be the cost of the predecessor label  $l'$  and assume that  $z'$  equals the cost of a path  $q'$  from  $s$  to  $v'$ . Then we have

$$z_0 = z'_0 + \hat{c}_{(v', v)} = \sum_{e \in q'} \hat{c}_e + \hat{c}_{(v', v)} = \sum_{e \in q' \cup (v', v)} \hat{c}_e.$$

For the other objectives we distinguish two cases:

- Case 1:  $\delta_{(v', v)} \leq z'_i \forall i = 1, \dots, \Gamma$ . In this case the  $\Gamma$  edges  $e$  with biggest intervals  $\delta_e$  of  $q'$  and  $q' \cup (v', v)$  are the same and  $z_i = z'_i$  for all objectives. Therefore,  $(z_0, \dots, z_\Gamma) = z^L(q \cup (v', v))$ .
- Case 2: Either  $\delta_{(v', v)} > z'_i$  for  $i = 1$  or  $\exists i \in \{2, \dots, \Gamma\}$  with  $z'_{i-1} \geq \delta_{(v', v)} > z'_i$ . Then

$$\begin{aligned}
\forall j < i : z_j &= z'_j \text{ and } j\text{-}\max_{e \in q'} \delta_e = j\text{-}\max_{e \in q' \cup (v', v)} \delta_e \\
\text{for } j = i : z_j &= \delta_{(v', v)} = j\text{-}\max_{e \in q' \cup (v', v)} \delta_e \\
\forall j : \Gamma \geq j > i : z_j &= z'_{j-1} = j\text{-}\max_{e \in q \cup (v', v)} \delta_e
\end{aligned}$$

It follows  $(z_0, \dots, z_\Gamma) = z^L(q' \cup (v', v))$ . □

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 [GBR06]. In our case, since we only look for a complete set of efficient solutions, the following weaker property is sufficient (this was observed but not proven in [IMP10]).

**Lemma 32.** *Let  $q$  from  $s$  to  $t$  be an efficient path with respect to  $z^L$  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  $p$  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. Then there exists an efficient path  $p$  from  $v$  to  $w$  that dominates  $q_{v,w}$ . We have

$$\sum_{e \in q'} \hat{c}_e = \sum_{e \in q_{s,v}} \hat{c}_e + \sum_{e \in p} \hat{c}_e + \sum_{e \in q_{w,t}} \hat{c}_e \leq \sum_{q_{s,v}} \hat{c}_e + \sum_{e \in q_{v,w}} \hat{c}_e + \sum_{e \in q_{w,t}} \hat{c}_e = \sum_{e \in q} \hat{c}_e$$

and for  $i = 1, \dots, \Gamma$  it follows from  $j\text{-}\max_{e \in p} \delta_e \leq j\text{-}\max_{e \in q_{v,w}} \delta_e$  that  $\forall j \leq i$

$$\begin{aligned} & i\text{-}\max_{e \in q'} \delta_e \\ &= i\text{-}\max \left( \{j\text{-}\max_{e \in q_{s,v}} \delta_e : j = 1, \dots, i\} \cup \{j\text{-}\max_{e \in p} \delta_e : j = 1, \dots, i\} \cup \{j\text{-}\max_{e \in q_{w,t}} \delta_e : j = 1, \dots, i\} \right) \\ &\leq i\text{-}\max \left( \{j\text{-}\max_{e \in q_{s,v}} \delta_e : j = 1, \dots, i\} \cup \{j\text{-}\max_{e \in q_{v,w}} \delta_e : j = 1, \dots, i\} \cup \{j\text{-}\max_{e \in q_{w,t}} \delta_e : j = 1, \dots, i\} \right) \\ &= i\text{-}\max_{e \in q} \delta_e. \end{aligned}$$

It follows  $z^L(q') \leq z^L(q)$  and we conclude  $z^L(q') = z^L(q)$ , because  $q$  is efficient with respect to  $z^L$ .  $\square$

**Theorem 33.** *When Algorithm 9 (with Algorithm 10 as Step 5) stops, the permanent labels at  $t$  represent a complete set of efficient solutions for Problem (3).*

*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 in [Ehr06] of the multi-objective label setting algorithm by Martins [Mar84]. For substituting the lexicographic order with the aggregate cost order see [IMP10].

Now, we show that the algorithm finds a complete set of efficient solutions. 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  $z = z^L(q)$ . Consider the predecessor node  $v'$  of  $t$  on  $q$ . From Lemma 32 it follows, that there is an efficient path  $p$  from  $s$  to  $v'$  with  $z^L(p \cup (v', t)) = z^L(q)$ .

If there exists a permanent label  $l'$  at  $v'$  with label costs  $z' = z^L(p)$ , then, at the moment when it was made permanent during the algorithm, a new label  $\bar{l}$  at node  $t$  with label costs  $\bar{z} = z' \oplus (\hat{c}_{(v',t)}, \delta_{(v',t)})$  would have been constructed. It follows

$$\bar{z} = z' \oplus (\hat{c}_{(v',t)}, \delta_{(v',t)}) = z^L(p) \oplus (\hat{c}_{(v',t)}, \delta_{(v',t)}) = z^L(p \cup (v', t)) = z^L(q).$$

Consider the first label with cost  $z^L(q)$  that was constructed at node  $t$ . If this label was deleted again, its costs are 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  $z^L(q)$  exists at  $t$ .

Therefore, there is no permanent label at the predecessor node  $v'$  of  $t$  with costs  $z'$  such that  $z' \oplus (\hat{c}_e, \delta_e) = z^L(q)$ . In the same way, we can show that there is no permanent label at the predecessor node  $v''$  of  $v'$  with costs  $z''$  such that

$$(z'' \oplus (\hat{c}_{(v'',v')}, \delta_{(v'',v')})) \oplus (\hat{c}_{(v',t)}, \delta_{(v',t)}) = z' \oplus (\hat{c}_{(v',t)}, \delta_{(v',t)}) = z^L(q).$$

By induction it follows that there is no permanent label at node  $s$  with cost  $(0, \dots, 0)$ , which is a contradiction, because such a label is constructed in Line 1 and made permanent during the first execution of Line 3.

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.  $\square$

To find a robust optimal solution (in the multi-objective case: a complete set of robust efficient solutions) we have to filter the solutions obtained by the labeling algorithm (see Algorithm 11).

---

**Algorithm 11** LSA for the shortest path problem with cardinality-constrained uncertainty

---

**Input:** an instance  $I = (E, Q, \hat{c}, \delta, \Gamma)$  of the (single-objective) cardinality-constrained uncertain shortest path problem

**Output:** a robust optimal solution for  $I$

- 1: Solve Problem (3) with Algorithm 9.
  - 2: Compare the aggregate cost of all permanent labels in  $t$  to find a minimal one.
  - 3: Obtain the path represented by this label by backtracking the predecessor labels.
- 

**Corollary 34.** *Algorithm 11 finds an optimal solution for (2) with non-negative edge costs.*

**Remark 35.** *The algorithms in this section can easily be extended for the multi-objective case. We will also use the abbreviation LSA to refer to the multi-objective version.*

### 3 Experiments

In this paper we have presented two approaches to solve multi-objective, cardinality-constrained uncertain combinatorial optimization problems. 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 are bottleneck functions. We have introduced such an algorithm for the shortest path problem (LSA) and, hence, we test our approaches on the multi-objective uncertain shortest path problem.

#### 3.1 Hazardous material transportation

We test our algorithms for the multi-objective uncertain shortest path problem 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. An overview about objectives for hazardous material transportation and about approaches for estimating the risk and the impacts of an accident is given in [ETV07].

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. Our problem instance for hazardous material transportation is based on the instance used in [KRSS16] to test an algorithm for bi-objective shortest path problems with only one uncertain objective. The underlying network is a sector of the Chicago region road network available at [BGKLS] (Chicago-regional). The sector contains 1301 nodes and 4091 edges.

To obtain plausible travel times in [KRSS16] a traffic assignment problem is solved 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 non-linear relationship between the flow on an edge and the travel time. Until an equilibrium

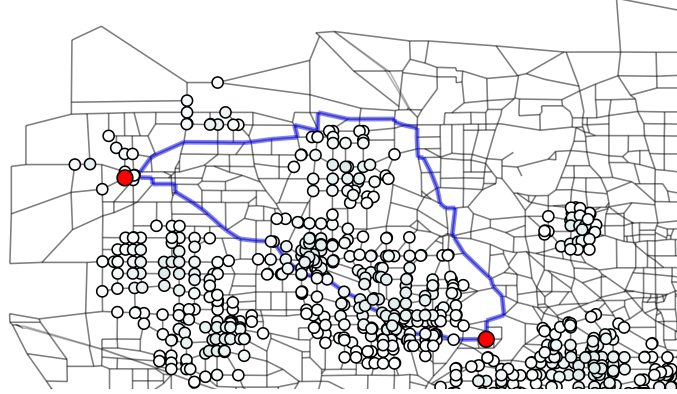


Figure 2: Section of the Chicago regional road network with distribution of population from [KRSS16]. The red dots show start and end node chosen for our experiments and two exemplary robust efficient paths are marked in blue.

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 in [KRSS16] 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 2 shows two exemplary robust efficient paths for the instance with  $x = 10$  and  $\Gamma = (5, 5)$ . 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.

### 3.2 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 2.1 and uses the special version DSA-oi for instances with objective independent element order (see Section 2.1.2). For solving the subproblems we use an implementation of the Algorithm of Martins [Mar84] (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 3.2.3, 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  $\Gamma$  (to keep the number of parameters low, we always choose the same value for  $\Gamma_1$  and  $\Gamma_2$  and we will refer to this value as  $\Gamma_i$  in the following). In addition, we compare the performance of the algorithms on an instance with three objectives and on an instance with two correlated objective functions. We further evaluate the improvement

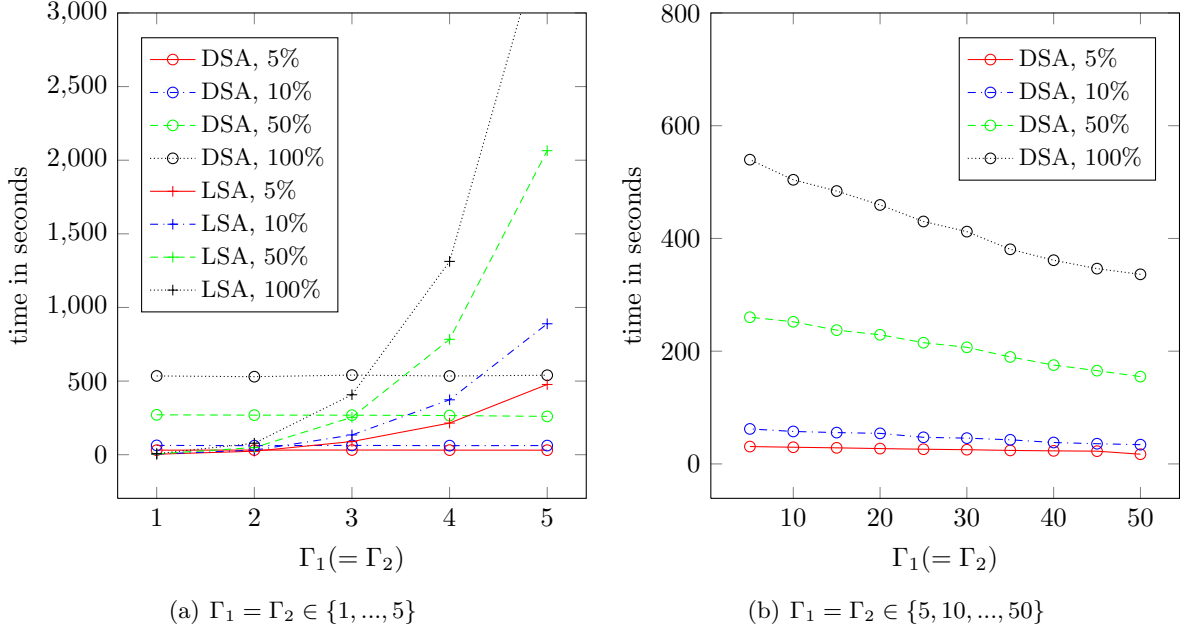


Figure 3: Running time of DSA and LSA for several values of  $\Gamma_i$  and population uncertainty  $x$  on two different scales.

gained by our enhancement of DSA (solution checking). Finally, we generate an instance with objective independent element order and compare the running time of DSA-oi for such instances to the general DSA.

### 3.2.1 Comparison of the two solution approaches for the hazardous material transportation instance

Figure 3 shows the running time of DSA and LSA for several values of  $\Gamma_i$  and  $x$ . The running time of LSA increases with  $\Gamma_i$ , whereas the running time of DSA decreases (see also Figure 6). The reason is that for increasing  $\Gamma_i$ , the number of objectives in the deterministic multi-objective problem solved during LSA increases as well, whereas the maximal number of subproblems solved during DSA decreases. For small values of  $\Gamma_i$  LSA solves the given instances faster, for higher values DSA has a better performance. If we choose a higher value for  $x$ , which results in a greater maximal and mean deviation from the nominal value and a higher number of different values of  $\delta_{e,2}$ , the running time of both algorithms increases. In the case of DSA, the increase of the running time can be explained by the higher number of different values of  $\delta_{e,2}$ , which leads to a higher number of subproblems.

### 3.2.2 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 using, again, the nominal population. We generate random interval lengths in the same range as the other population objective, i.e., the value of population uncertainty in general is the same for both population objectives, but the specific interval lengths of each edge may differ. Figure 4 shows the running times on this instance in comparison to the instance with two objectives described above.



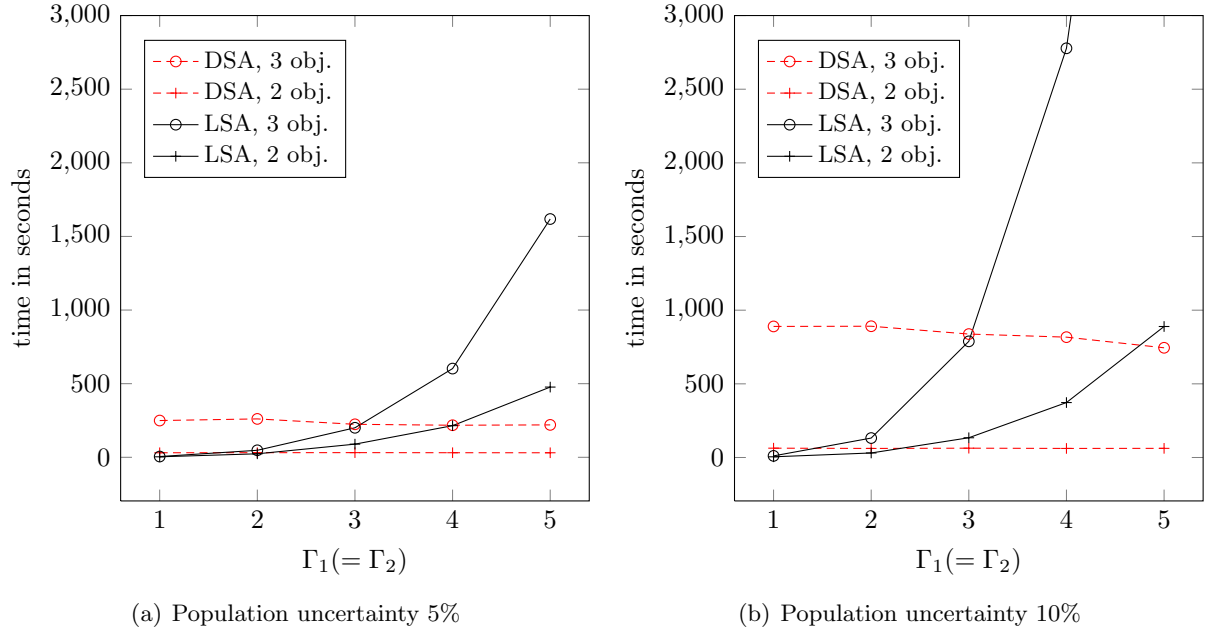


Figure 4: Running time of DSA and LSA for an instance with three objectives and an instance with two objectives.

The running time of both algorithms increases by including the additional objective. The relative difference between the running time of the instance with two objectives and the instance with three objectives increases with  $\Gamma_i$  for LSA, whereas it decreases for DSA.

### 3.2.3 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 5. In comparison, LSA benefits more from correlated objective function values: The values of  $\Gamma_i$ , 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 3.2.1. For small values of  $\Gamma_i$  it is much faster than DSA.

### 3.2.4 Evaluation of the improvement obtained by solution checking

To evaluate the obtained improvement by using solution checking in DSA, we use Algorithm 8 as Step 3 of Algorithm 6. We compare the running time of the version containing solution checking to the running time of the version without this enhancement, and, additionally, count the solved subproblems (Figure 6). Where fewer subproblems have been 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  $\Gamma_i$  increases solution checking becomes less effective.

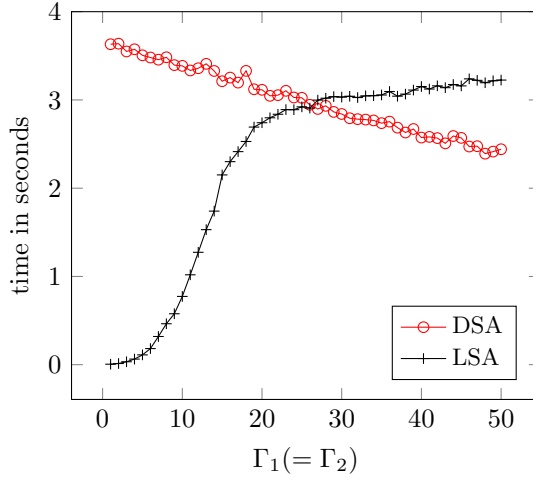


Figure 5: Running time of DSA and LSA for an instance with two strongly correlated objective functions.

Note that, since Lemma 21 allows to exclude even more subproblems than excluded in Algorithm 8, further speed-ups may be achieved by implementing a more sophisticated solution checking. However, already when using Algorithm 8, the benefit of solution checking is clearly visible.

### 3.2.5 Evaluation of DSA for instances with objective independent element order

To compare the performance of the objective independent DSA (DSA-oi) from Section 2.1.2 to the general algorithm, we generate 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 7 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.

## 4 Conclusion

In this paper we have developed two approaches to find minmax robust solutions for multi-objective combinatorial optimization problems with cardinality-constrained uncertainty. We have extended an algorithm from [BS03] (DSA) to multi-objective optimization, have suggested an enhancement and developed a special version for instances with objective independent element order. We have also introduced a second approach and used it to develop a label setting algorithm (LSA) for the multi-objective uncertain shortest path problem.

We have 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  $\Gamma_i$ . If the two objective functions are strongly correlated, which appears often in shortest path problems, where, e.g., the distance, travel time and fuel consumption are correlated, LSA is competitive even for higher values of  $\Gamma_i$ .

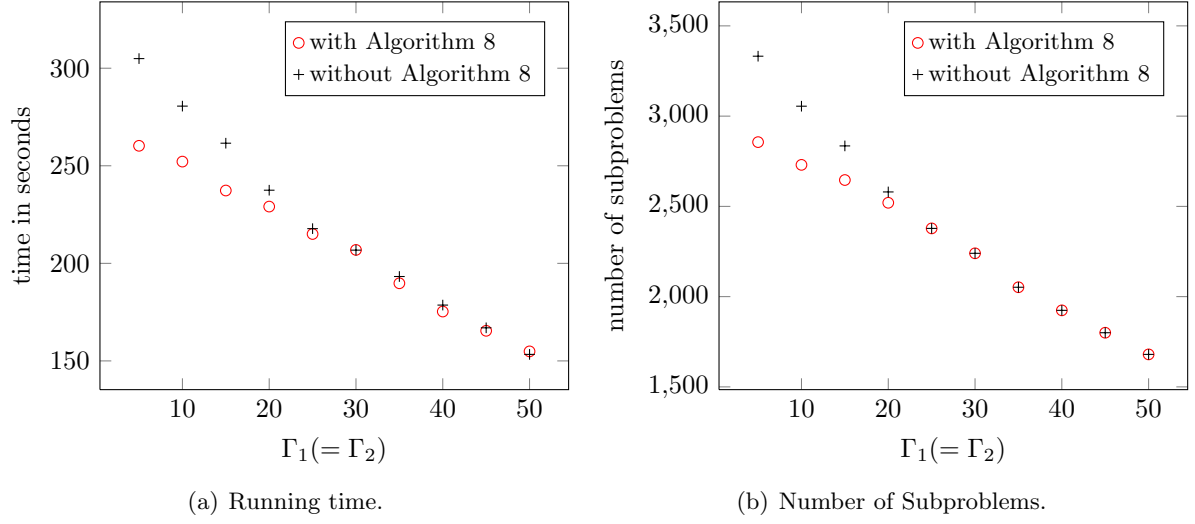


Figure 6: Running time and number of solved subproblems of DSA with and without solution checking (Population uncertainty 50%).

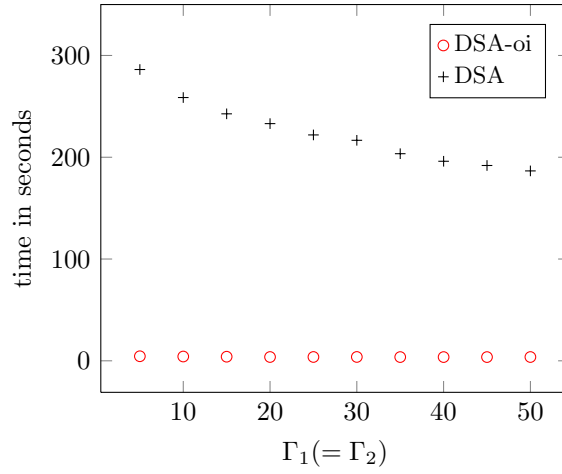


Figure 7: Comparison of DSA and DSA-oi for instances with objective independent element order.

When implementing DSA we recommend to use the proposed enhancements and to check whether the special version for instances with (partly) 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 can not be realized exactly, is of interest, see [EKS15] 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*, where a complete set of efficient paths from a start node  $s$  to all other nodes has to be found. Since, in the deterministic case, there exist algorithms (e.g. the algorithm of Martins [Mar84]) 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

- [ABV09] H. Aissi, C. Bazgan, and D. Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009.
- [BGKLS] H. Bar-Gera, C. Kwon, J. Li, and B. Stabler. Transportation networks. <https://github.com/bstabler/TransportationNetworks>. Accessed: 2016-11-04.
- [BS03] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical programming*, 98(1):49–71, 2003.
- [BS04] D. Bertsimas and M. Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.
- [Chu16] T. D. Chuong. Optimality and duality for robust multiobjective optimization problems. *Nonlinear Analysis: Theory, Methods & Applications*, 134:127–143, 2016.
- [DKW12] E.K. Doolittle, H.L. M. Kerivin, and M. M. Wiecek. A robust multiobjective optimization problem with application to internet routing. Technical Report R2012-11-DKW, Clemson University, 2012.
- [Ehr06] M. Ehrgott. *Multicriteria optimization*. Springer, Berlin, Heidelberg, 2006.
- [EIS14] M. Ehrgott, J. Ide, and A. Schöbel. Minmax robustness for multi-objective optimization problems. *European Journal of Operational Research*, 239:17–31, 2014.

- [EKS15] G. Eichfelder, C. Krüger, and A. Schöbel. Multi-objective regularization robustness. Technical Report 2015-13, Preprint-Reihe, Institut für Numerische und Angewandte Mathematik, Georg-August Universität Göttingen, 2015.
- [ETV07] E. Erkut, S. A. Tjandra, and V. Verter. Hazardous materials transportation. *Handbooks in operations research and management science*, 14:539–621, 2007.
- [FW14] J. Fliege and R. Werner. Robust multiobjective optimization & applications in portfolio optimization. *European Journal of Operational Research*, 234(2):422–433, 2014.
- [GBR06] X. Gandibleux, F. Beugnies, and S. Randriamasy. Martins’ algorithm revisited for multi-objective shortest path problems with a maxmin cost function. *4OR*, 4(1):47–59, 2006.
- [GKR12] J. Gorski, K. Klamroth, and S. Ruzika. Generalized multiple objective bottleneck problems. *Operations Research Letters*, 40(4):276–281, 2012.
- [GS16] M. Goerigk and A. Schöbel. Algorithm engineering in robust optimization. In L. Kliemann and P. Sanders, editors, *Algorithm Engineering: Selected Results and Surveys*, volume 9220 of *LNCS State of the Art*, pages 245–279. 2016.
- [HNS13] F. Hassanzadeh, H. Nemati, and M. Sun. Robust optimization for multiobjective programming problems with imprecise information. *Procedia Computer Science*, 17:357 – 364, 2013.
- [IKK<sup>+</sup>14] J. Ide, E. Köbis, D. Kuroiwa, A. Schöbel, and C. Tammer. The relationship between multi-objective robustness concepts and set valued optimization. *Fixed Point Theory and Applications*, 2014(83), 2014.
- [IMP10] M. Iori, S. Martello, and D. Pretolani. An aggregate label setting policy for the multi-objective shortest path problem. *European Journal of Operational Research*, 207(3):1489–1496, 2010.
- [IS16] J. Ide and A. Schöbel. Robustness for uncertain multi-objective optimization: A survey and analysis of different concepts. *OR Spectrum*, 38(1):235–271, 2016.
- [ITWH15] J. Ide, M. Tiedemann, S. Westphal, and F. Haiduk. An application of deterministic and robust optimization in the wood cutting industry. *4OR*, 13(1):35–57, 2015.
- [KDD16] M. Kalantari, C. Dong, and I. J. Davies. Multi-objective robust optimisation of unidirectional carbon/glass fibre reinforced hybrid composites under flexural loading. *Composite Structures*, 138:264–275, 2016.
- [KL12] D. Kuroiwa and G. M. Lee. On robust multiobjective optimization. *Vietnam J. Math*, 40(2-3):305–317, 2012.
- [KRSS16] K. Kuhn, A. Raith, M. Schmidt, and A. Schöbel. Bicriteria robust optimization. *European Journal of Operational Research*, 252:418–431, 2016.
- [LK14] T. Lee and C. Kwon. A short note on the robust combinatorial optimization problems with cardinality constrained uncertainty. *4OR*, 12(4):373–378, 2014.
- [Mar84] E. Q. V. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236–245, 1984.
- [PL07] K.-C. Park and K.-S. Lee. A note on robust combinatorial optimization problem. *Management Science and Financial Engineering*, 13(1):115–119, 2007.
- [WD16] M. M. Wiecek and G. M. Dranichak. Robust multiobjective optimization for decision making under uncertainty and conflict. In *Optimization Challenges in Complex, Networked and Risky Systems*, pages 84–114. INFORMS, 2016.
- [YL13] H. Yu and H. M. Liu. Robust multiple objective game theory. *Journal of Optimization Theory and Applications*, 159(1):272–280, 2013.