Ranking Robustness and its Application to Evacuation Planning*

Marc Goerigk[†1], Horst W. Hamacher[2], and Anika Kinscherff[2]

[1]Department of Management Science, Lancaster University, United Kingdom
[2]Fachbereich Mathematik, Technische Universität Kaiserslautern, Germany

Abstract

We present a new approach to handle uncertain combinatorial optimization problems that uses solution ranking procedures to determine the degree of robustness of a solution. Unlike classic concepts for robust optimization, our approach is not purely based on absolute quantitative performance, but also includes qualitative aspects that are of major importance for the decision maker.

We discuss the two variants, solution ranking and objective ranking robustness, in more detail, presenting problem complexities and solution approaches. Using an uncertain shortest path problem as a computational example, the potential of our approach is demonstrated in the context of evacuation planning due to river flooding.

Keywords: Robust Optimization, Solution Ranking, Combinatorial Optimization, Evacuation Planning

1 Introduction

In the recent past, several evacuations became necessary due to river flooding in various parts of the world. From a planning point of view, operations research methods have a high potential to be used quite successfully in this context (see, for instance, [HT01]), since there is usually some time before the decision for an evacuation is made and the actual evacuation is started. Obviously, the water level in flooded areas is dependent on the rain fall causing the flooding, and the latter is subject to uncertainty. Therefore, robust optimization models are very appropriate to deal with flood evacuation.

Since its first formalization in the late 90s, robust optimization has seen uninterrupted rising interest both from the research community as well as from practitioners. Following the seminal work [BTN98], many different variants have evolved, each catering to the specialized needs of some application, or a better

*Partially supported by the German Ministry of Research and Technology (BMBF), projects RobEZiS, grant number 13N13198 and StanLay, grant number 13N12826, and by the Air Force Office of Scientific Research, Air Force Material Command, USAF, grant number FA8655-13-1-3066. The U.S Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright notation thereon.

[†]Email: m.goerigk@lancaster.ac.uk, {hamacher,kinscherff}@mathematik.uni-kl.de

trade-off between conservatism and costs. We refer to [BBC11, BTGN09, GS15] for surveys on the topic, and to [GYdH15, CG16b] for more hands-on guides on robust optimization.

In this paper we focus on combinatorial optimization problems with uncertain cost coefficients. As a typical example, consider a shortest path problem in a road network, where the time to traverse an edge is not known exactly, and even no probability distribution is available. More formally, we write

$$P(c) \quad \min\{f(x,c) \ : \ x \in \mathcal{X}\}, \quad c \in \mathcal{U} \tag{1}$$

where $\mathcal{X}$ denotes the set of feasible solutions, and $\mathcal{U}$ a set of possible scenarios, the so-called uncertainty set.

As noted above, there exist many approaches to reformulate this family of problems $P(c)$ to a robust counterpart, whose optimal solution should perform "well" over all possible scenarios in some sense that needs to be specified. For this type of problems, we refer to the overview [ABV09].

In this paper, we restrict ourselves to two classical robust counterparts. The first one, *minmax robustness* (also known as strict robustness)

$$MM \quad \min \left\{ \max_{c \in \mathcal{U}} f(x,c) \ : \ x \in \mathcal{X} \right\} \tag{2}$$

is a conservative measure based on the absolute objective values of all scenarios. The second one uses a relative measure comparing objective values of a given solution with the best possible one and is known as *minmax regret*:

$$MMR \quad \min \left\{ \max_{c \in \mathcal{U}} (f(x,c) - f^*(c)) \ : \ x \in \mathcal{X} \right\}. \tag{3}$$

Here $f^*(c) := \min\{f(x,c) : x \in \mathcal{X}\}$ is the best possible objective value with respect to scenario $c \in \mathcal{U}$ and is used as a benchmark for any other solution $x \in \mathcal{X}$. Both approaches evaluate the robustness of a solution only based on its (absolute or relative) worst-case performance in the objective.

An *ideal minmax regret solution* $x^I \in \mathcal{X}$ is one, where the objective value

$$\max_{c \in \mathcal{U}} \ \big(f(x^I,c) - f^*(c)\big) = 0 \tag{4}$$

of *MMR* is equal to 0, which means that some $x^I \in \mathcal{X}$ can be found which is optimal for each scenario $c \in \mathcal{U}$. Although an ideal minmax regret solution is highly desirable, one can, in general, not expect to find such a solution. We, therefore, propose in this paper a modified version, the *ranking robust counterpart* which relaxes the condition of an ideal minmax regret solution to

$$\max_{c \in \mathcal{U}} \ \big(f(x^{RR},c) - f(x^K(c),c)\big) = 0 \tag{5}$$

where $x^K(c)$ is a K best solution of $P(c)$ in (1).

As a numerical example, consider the following minimization problem with two scenarios c_1 and c_2, and three solutions A, B, and C. The objective values are given in Table 1.

Solution A has the best worst-case performance, and is the optimal solution to *MM*. However, it ignores the poor performance of A compared to B and C

	A	B	C
c_1	50	21	10
c_2	100	105	110

Table 1: Objective values of an example problem.

in scenario c_1. Solution C has the smallest maximum regret, and is the optimal solution to *MMR*. Solution B is the second-best solution in every scenario, and is thus also interesting as a compromise solution from a practical perspective (while both A and C can be the worst choices in one of the scenarios, respectively).

Alternatively, we may also consider each scenario as an objective function of a multi-criteria optimization problem. There usually does not exist a single solution that performs best for all objective functions at the same time; instead, one aims at finding Pareto solutions (see [Ehr06]). It can be shown that the set of Pareto solutions also includes optimal solutions to *MM* and *MMR* [ABV09].

Choosing one solution out of the set of Pareto solutions is already a difficult task that is hard to automate, as it depends on the practical insight and priorities of the decision maker (see [Mie14] for a survey on visualization methods that guide such a selection process). One approach to select such a desired solution from the set of candidates is to roughly classify their performance in each objective, and to choose one that never falls into a bottom-percentile performance class. Such an approach also leads to our concept of ranking robustness.

Our method is related to the robust optimization approach presented in [BMSW13]. For any $\rho \geq 1$, the authors consider the set of ρ-approximate solutions in each scenario. Their aim is to find ρ large enough, such that the intersection of these sets is non-empty. Furthermore, a value for ρ is to be found which maximizes what they call the unexpected similarity between the solution sets.

In the following, we formalize our approach of ranking robustness. We introduce a general definition for ranking robust optimization problems and discuss general properties in Section 2. We then consider two variants in more detail: Solution Ranking Robustness in Section 3, and Objective Ranking Robustness in Section 4. These approaches are compared in Section 5, and applied to the shortest path problem in Section 6. A computational example applying our approach to the shortest path problem on a real-world street network in the context of evacuation planning is presented in Section 7, before the paper is concluded in Section 8.

2 Ranking Robustness

We consider combinatorial optimization problems

$$(P) \qquad \min\{f(x,c) = c^t x : x \in \mathcal{X}\} \tag{6}$$

over some set $\mathcal{X} \subseteq 2^E$ of feasible solutions, where $E = \{e_1, \ldots, e_m\}$ is a finite ground set (equivalently $\mathcal{X} \subseteq \mathbb{B}^m$ and $x \in \mathcal{X}$ a binary vector). Due to data uncertainty, we assume that the cost coefficients c are not known exactly, but are known to stem from some set of possible outcomes $\mathcal{U}$, also called the uncertainty set. We write $P(c)$, $c \in \mathcal{U}$ to denote that problem P is uncertain and depending on c.

Inspired by ranking problems (often also referred to as K best problems, see, e.g., [HQ85]), we introduce the following notation.

Definition 1. *For each $c \in \mathcal{U}$ a **priority list (with respect to c) with length $L(c)$** is an ordered partition of the set $\mathcal{X}$ of feasible solutions into $L(c)$ subsets, i.e.,*

$$\mathcal{S}(c) = \big(S_1(c), S_2(c), \ldots, S_{L(c)}(c)\big) \;\; \textit{with} \;\; \bigcup_{i=1}^{L(c)} S_i(c) = \mathcal{X}, \;\; S_i(c) \cap S_j(c) = \emptyset$$

for $i \neq j, i, j \in \{1, ..., L(c)\}$.

Definition 2. *Given $c \in \mathcal{U}$ and a priority list $\mathcal{S}(c)$, $x \in S_i(c)$ is said to be **preferred to** $y \in S_j(c)$ iff $i < j$. For $x \in S_i(c)$ and $y \in S_j(c)$ with $i < j$, we say that x is preferred to y in scenario $c \in \mathcal{U}$.*

Generally speaking, a priority list should encapsule the preferences of a decision maker under each scenario. Hence, there may be different approaches to construct such lists. We illustrate some in the following.

Example 1. *We consider the shortest $s-t$ path instance from Figure 1. Next*

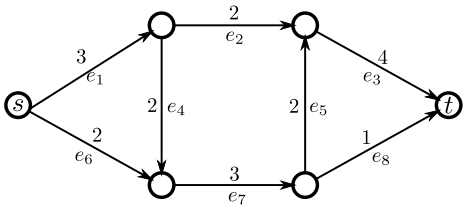

Figure 1: A shortest path instance.

to each edge, its name and length are shown. Table 2 summarizes all feasible solutions in this setting.

Path Name	Path	Length
P_1	(e_1, e_2, e_3)	9
P_2	$(e_1, e_4, e_7, e_5, e_3)$	14
P_3	(e_1, e_4, e_7, e_8)	9
P_4	(e_6, e_7, e_5, e_3)	11
P_5	(e_6, e_7, e_8)	6

Table 2: Feasible solutions to the example shortest path instance from Figure 1.

One natural approach to construct a priority list is to group paths according to their objective ranking, i.e., their total length, which yields

$$\mathcal{S}^{OR}(c) = (\{P_5\}, \{P_1, P_3\}, \{P_4\}, \{P_2\}) \tag{7}$$

Note that paths P_1 and P_3 have the same length, and are therefore given the same level or priority. Alternatively, we may decide that every priority class

should consist of a single solution, e.g., by applying some lexicographic quality criterion. If we use the number of edges in a path to refine the set $\{P_1, P_3\}$ *further, we get*

$$\mathcal{S}^{SR}(c) = (\{P_5\}, \{P_1\}, \{P_3\}, \{P_4\}, \{P_2\}). \tag{8}$$

As we will see later, priority lists consisting of singletons have algorithmic advantages. Finally, it is even possible to use priority lists which are not solely based on objective values, but on additional expertise of the planner; as an example, we assume that there exists some priority order on the set of edges E*, such that edges are preferred in the order* $(e_8, e_3, e_1, e_6, e_4, e_5, e_2, e_7)$*. Following such an approach, we find*

$$\mathcal{S}^{PL}(c) = (\{P_3\}, \{P_5\}, \{P_2\}, \{P_1\}, \{P_4\}).$$

In this paper, we focus on priority lists of type $\mathcal{S}^{OR}(c)$ and $\mathcal{S}^{SR}(c)$ which we formalize subsequently.

Definition 3. *A priority list* $\mathcal{S}(c)$ *is called* ***objective-ranking****, if*

$$f(x,c) < f(y,c) \text{ for all } x \in S_i(c), y \in S_j(c) \text{ with } i < j \tag{9}$$

and

$$f(x,c) = f(y,c) \text{ for all } x, y \in S_i(c). \tag{10}$$

It is called ***solution-ranking*** *if*

$$f(x,c) \leq f(y,c) \text{ for all } x \in S_i(c), y \in S_j(c) \text{ with } i < j \tag{11}$$

and

$$|S_i(c)| = 1 \text{ for all } i = 1, \ldots, L(c). \tag{12}$$

Based on these definitions, we now consider their usage in robust optimization. To this end, we consider solutions that guarantee a certain preference over all scenarios $c \in \mathcal{U}$.

Definition 4. *Let a priority list* $\mathcal{S}(c)$ *be given for every scenario* $c \in \mathcal{U}$*, and let* $K \in \mathbb{N}$*. Then we denote with*

$$\mathcal{X}^K(c) := \bigcup_{i \leq K} S_i(c) \tag{13}$$

the set of feasible solutions with preference at most K *in scenario* c*, and with*

$$\mathcal{X}^K := \bigcap_{c \in \mathcal{U}} \mathcal{X}^K(c). \tag{14}$$

Here we assume $S_i(c) = \emptyset$ for $i > L(c)$ such that $\mathcal{X}^K(c)$ is well-defined for all $K \in \mathbb{N}$ in (13). We are now in a position to define our new approach to robust optimization, ranking robustness.

Definition 5. *We say a solution* $x \in \mathcal{X}$ *is* ***K-ranking robust*** *if* $x \in \mathcal{X}^K$*. The* ***(general) ranking robustness problem*** *(RR) consists in finding* $K^* := \min\{K \in \mathbb{N}_+ : \mathcal{X}^K \neq \emptyset\}$*, i.e. the smallest* K *for which a* K*-ranking robust solution exists. Given a solution- or objective-ranking, the corresponding RR problems are called* ***Solution-Ranking Robustness*** *(SRR) and* ***Objective-Ranking Robustness*** *(ORR), respectively.*

It should be noted that any solution $x^{RR} \in \mathcal{X}^{K^*}$ satisfies the relaxed ideal minmax regret robustness criterion presented in (5). A small K^* can be considered as an indicator of a well-conditioned uncertainty set, since one can find a feasible solution which is close to an ideal minmax regret solution. Large K^* indicate a large variability in the scenarios.

We illustrate the ranking robustness approach by extending Example 1.

Example 2. *Let us denote the scenario shown in Figure 1 as c_1, and let us assume there exists a second scenario c_2 with $c_2 = (1, 3, 1, 1, 4, 4, 2, 3)$. Then the objective ranking for c_2 is given as*

$$\mathcal{S}^{OR}(c_2) = (\{P_1\}, \{P_3\}, \{P_2, P_5\}, \{P4\})$$

and the solution ranking as

$$\mathcal{S}^{SR}(c_2) = (\{P_1\}, \{P_3\}, \{P_5\}, \{P_2\}, \{P4\})$$

Using the priority lists (7,8) of $c = c_1$ we find that for objective ranking, $K^* = 2$ with both P_1 and P_3 being 2-ranking robust, while for solution ranking, $K^* = 2$ with only P_1 being 2-ranking robust.

When the priority lists for all scenarios are given, problem RR can be solved by iteratively testing if the intersections $\cap_{c \in \mathcal{U}} \mathcal{X}^K(c)$ are empty. This can be done in polynomial time in the cardinality of $\mathcal{X}$ (as the priority lists are part of the input, this means a polynomial solution time overall).

In general the solution set $\mathcal{X}^{K^*}$ contains more than one solution which are all considered as optimal for RR. Hence, if a single solution should be presented to the decision maker, we are facing the problem which solution from $\mathcal{X}^{K^*}$ to choose (naturally, this also applies to most other robust optimization concepts, such as *MM* and *MMR*). There are several selection criteria conceivable, some of which are described below.

- Choose the best solution with respect to the nominal scenario (if existent).
- Choose the best solution with respect to the minmax problem.
- Choose the best solution with respect to the minmax regret problem.
- Choose the solution minimizing the mean objective value over all scenarios.
- Apply another existing robustness concept on $\mathcal{X}^{K^*}$ instead of on $\mathcal{X}$.

Choosing a specific criterion, say criterion $\mathcal{C}$, we apply $\mathcal{C}$ on the solution set $\mathcal{X}^{K^*}$ to obtain the final optimal solution of the problem using the RR concept. If there is more than one solution which fulfills $\mathcal{C}$, we choose one arbitrarily.

Note that for both SRR and ORR, solutions in $\mathcal{X}^1$ are also optimal for *MM* and *MMR*, if they exist.

For a more in-depth discussion of solution postprocessing in robust optimization, we refer to [IT14].

3 Solution-Ranking Robustness

In this section we consider the SRR problem first for finite uncertainty sets $\mathcal{U}$, and then show that the problem of interval uncertainty sets can be reduced to finite ones.

In SRR, the cardinality constraint (12) implies $|\mathcal{X}^K(c)| = K$ for all $K \in \mathbb{N}$, and $\mathcal{S}(c)$ can be written for all $c \in \mathcal{U}$ as

$$\mathcal{S}(c) = \left(x^1(c), x^2(c), \ldots, x^{L(c)}(c)\right), \tag{15}$$

where $x^k(c)$ is the k-th best solution of problem $P(c), k = 1, \ldots, L(c)$ and $L(c) = |\mathcal{X}|$.

In contrast to the ORR problem, the ordering inequality (11) is, in general, not strict, i.e., it may happen that $f(x^i(c), c) = f(x^j(c), c)$ for $i \neq j$. Hence, to specify the solution priority lists, we assume the existence of a common tie breaking rule for all $c \in \mathcal{U}$ to decide a consistent ordering in $\mathcal{U}$:

Assumption 1. *If two solutions x and y have the same objective value with respect to $c \in \mathcal{U}$ and x is preferred to y in this scenario, then x is also preferred to y in every other scenario $c' \in \mathcal{U}$ with $f(x, c') = f(y, c')$.*

Due to Definition 5 and (15), a feasible solution satisfies $x \in \mathcal{X}^K$ iff it is among the K best solutions for every scenario $c \in \mathcal{U}$. We can thus apply any ranking algorithm for combinatorial optimization problems to compute $\mathcal{X}^K(c) = \left(x^1(c), x^2(c), \ldots, x^K(c)\right)$, for instance, the binary search tree (BST) algorithm of [HQ85]. We sketch this algorithm to keep the paper self-contained.

For the BST procedure we assume to have an algorithm computing in addition to the best also a second best feasible solution of $P(c)$. In the initialization step, this algorithm is applied to the whole feasible set $\mathcal{X}$ and returns $x^1 = x^{0,1}$ and $x^2 = x^{0,2}$, respectively. Then $\mathcal{X}$ is partitioned into two disjoint sets $\mathcal{X}_1$ and $\mathcal{X}_2$ such that x^1 is the best solution of $\mathcal{X}_1$ and x^2 is the best solution of $\mathcal{X}_2$. We can choose these sets using an element $e \in E$ such that $x^1_e = 1$ and $x^2_e = 0$ by setting

$$\mathcal{X}_1 := \mathcal{X} \cap I, \text{ where } I := \{x \in X : x_e = 1\} \tag{16}$$

and

$$\mathcal{X}_2 := \mathcal{X} \cap O, \text{ where } O := \{x \in X : x_e = 0\} \tag{17}$$

For both sets the second best solutions $x^{1,2}$ (in $\mathcal{X}_1$) and $x^{2,2}$ (in $\mathcal{X}_2$) are computed and compared. Assuming w.l.o.g. that $x^{2,2}$ is better, we find the third best solution $x^3 := x^{2,2}$ for the original feasible set $\mathcal{X}$.

$\mathcal{X}_2$ is replaced by its partition $\mathcal{X}_3$ and $\mathcal{X}_4$, with $x^{3,1}$ and $x^{4,1}$ being the corresponding best solutions. These steps are repeated until we receive the K-th best solution.

Figure 2 illustrates how to find the three best solutions.

The complexity of this algorithm to find the K best solutions of $P(c)$ is $\mathcal{O}(B(m) + (K-1)C(m))$, where $B(m)$ and $C(m)$ denote the computational effort to determine the best and restricted second best solution, respectively. Since in the worst case $K^* = |\mathcal{X}|$ is possible for the SRR problem, this yields an $\mathcal{O}(|\mathcal{U}|(B(m) + |\mathcal{X}|C(m)))$ algorithm.

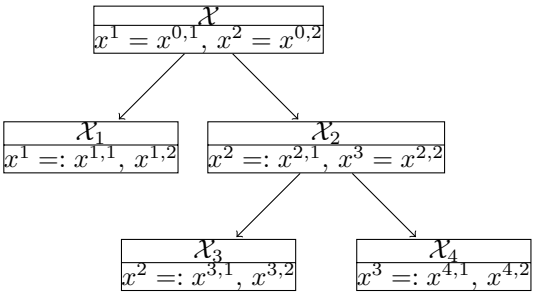

Figure 2: *BST* approach with computed 3-best solutions. The 4-th best is the best of $x^{1,2}, x^{3,2}$ and $x^{4,2}$.

In order to avoid this exponential running time, we suggest to combine the ranking approach with a relaxation of the ideal minmax regret equation (4): Fix some scenario $c \in \mathcal{U}$ and some $K \in \mathbb{N}$, compute $\mathcal{X}^K(c) = \{x^1, ..., x^K\}$ and the value

$$\min_{i=1,\dots,K} \max_{c \in \mathcal{U}} \left(f(x^i, c) - f^*(c) \right). \tag{18}$$

The solution x^i minimizing (18) is used as approximate solution. This approach generalizes [MG04] from shortest paths to other combinatorial optimization problems (see [HQ85]), and has a complexity of $\mathcal{O}(B(m)+(K-1)C(m)+K|\mathcal{U}|)$. Alternatively, one could stop the *SRR* algorithm, if the bound (18) is smaller than a given accuracy ϵ.

We conclude this section by proving that we can solve a *SRR*-problem for a special case of infinite uncertainties the same way as for finite uncertainties.

Theorem 1. *Let Assumption 1 hold and $\mathcal{U}$ be an interval-based uncertainty set, i.e., $c_e \in [\underline{c}_e, \overline{c}_e]$ for all $e \in E$, and let $Ext(\mathcal{U})$ be the extreme points of $\mathcal{U}$. Then $x \in \mathcal{X}$ is K-solution ranking robust with respect to $\mathcal{U}$ if and only if x is K-solution ranking robust with respect to $Ext(\mathcal{U})$.*

Proof. Since $Ext(\mathcal{U}) \subseteq \mathcal{U}$, any solution that is K-solution ranking robust w.r.t $\mathcal{U}$ is also K-solution ranking robust w. r. t. $Ext(\mathcal{U})$.

To prove the converse, suppose that $x \in \mathcal{X}$ is not K-solution ranking robust with respect to $\mathcal{U}$. Then there exists a scenario $c \in \mathcal{U}$ and solutions $y^k \in S_k(c)$ for $k = 1, \dots, K$ that are preferred to x. As we consider solution rankings, it holds that

$$\sum_{e \in E} c_e x_e \geq \sum_{e \in E} c_e y_e^K \geq \dots \geq \sum_{e \in E} c_e y_e^1$$

We define the following scenario $\hat{c} \in Ext(c)$:

$$\hat{c}_e = \begin{cases} \overline{c}_e, & if \quad x_e = 1 \\ \underline{c}_e, & if \quad x_e = 0 \end{cases}$$

By construction of $\hat{c}$, it holds that

$$\sum_{e \in E} \hat{c} x_e \geq \sum_{e \in E} c_e x_e.$$

For any $k \in \{1, \dots, K\}$, we therefore have that

$$\begin{aligned}\sum_{e\in E} \hat{c}_e x_e - \sum_{e\in E} \hat{c}_e y_e^k &= \sum_{\substack{e\in E\\ x_e=1, y_e^k=0}} \overline{c}_e - \sum_{\substack{e\in E\\ x_e=0, y_e^k=1}} \underline{c}_e \\ &\geq \sum_{\substack{e\in E\\ x_e=1, y_e^k=0}} c_e - \sum_{\substack{e\in E\\ x_e=0, y_e^k=1}} c_e \\ &= \sum_{e\in E} c_e x_e - \sum_{e\in E} c_e y_e^k \;\geq\; 0\end{aligned}$$

Due to Assumption 1, this means that y^k is preferred to x also in scenario $\hat{c}$. Hence, x cannot be K-solution ranking robust with respect to $Ext(\mathcal{U})$. □

4 Objective-Ranking Robustness

For ORR we can generate the priority lists using an adaption of the BST ranking algorithm, but since the number of solutions with the same objective value can be up to $|\mathcal{X}|$, we develop another way to solve it. For $i = 1, ..., L(c)$ let $val(i, c) := f(x, c)$ be the unique common objective value within each set $S_i(c)$ (recall Definition 3). For ease of notation we set $val(i, c) = val(L(c), c)$ for $i > L(c)$.

Suppose the first N values of $val(i, c)$ are known for all $c \in \mathcal{U}$, and $K^* \leq N$. Then the ORR problem can be formulated as the following integer program (IP):

$$\min\ K \tag{19}$$

$$\text{s.t. } c^t x \leq val(i, c) + M(c) z_i \qquad \forall c \in \mathcal{U}, i = 1, \dots, N \tag{20}$$

$$\sum_{i=1}^{N} z_i \leq (K - 1) \tag{21}$$

$$x \in \mathcal{X} \tag{22}$$

$$z_i \in \{0, 1\} \tag{23}$$

$$K \in \mathbb{N}_+ \tag{24}$$

where $M(c)$ is a constant sufficiently large (e.g. $M(c) \geq \sum_{e\in E} c_e$) and z_i a variable equal to 1 whenever there is no x which is i-ranking robust. If solving this problem shows that $K^* > N$, we need to increase N by determining additional values $val(i, c)$. Finding $val(i, c)$ given $val(i - 1, c)$ can be done also using an IP, or by using the BST algorithm from Section 3.

If we only want to check if there exists a K-objective ranking robust solution given the first K values of $val(i, c)$, we solve the problem

$$\min_{x\in\mathcal{X}} \max_{c\in\mathcal{U}} (f(x, c) - val(K, c))$$

instead. Note the similarity of this formulation to problem *MMR*.

The next example shows that for the ORR approach – in contrast to SRR – it is not possible to find a solution by restricting an uncertainty set to its extreme points, even in the case of intervals.

Example 3. *Consider an uncertain shortest path problem with three disjunct paths P_1, P_2 and P_3 from s to t, with costs $c_1 \in [5,9]$, $c_2 \in [6,13]$ and $c_3 \in [6,14]$. There are eight extreme scenarios; solving ORR using only these gives $K^* = 2$ and $\mathcal{X} = \{P_1\}$. However, in the scenario $c_1 = 8$, $c_2 = 6$ and $c_3 = 7$, we find that $P_1 \notin \mathcal{X}^2(c)$.*

To analyze the complexity of *ORR*, we consider the unconstrained combinatorial optimization problem

$$\text{(UP)} \qquad \min \left\{ \sum_{e \in E} c_e x_e \mid x \in \{0,1\}^m \right\}$$

in the following. Note that the minmax robust counterpart of (UP) is NP-hard already for two scenarios and unrestricted c_e (see [BBI14]).

We show that calculating $val(i,c)$ for some fixed $c \in \mathcal{U}$ and given $val(i-1,c)$ is NP-hard. An integer program to calculate $val(i,c)$ in the case of $c \in \mathbb{N}^m$ is the following.

$$\begin{aligned} \text{(UP-val)} \qquad \min \ & \sum_{e \in E} c_e x_e \\ \text{s.t.} \ & \sum_{e \in E} c_e x_e \geq val(i-1,c) + 1 \\ & x \in \mathbb{B}^m \end{aligned}$$

Theorem 2. *Problem (UP-val) is NP-hard.*

Proof. The decision problem of (UP-val) is to decide whether there exists a solution x such that $K_1 \leq \sum_{e \in E} c_e x_e \leq K_2$ for given values K_1, K_2.

We use a reduction from the Partition problem. Given a set A and weights $w : A \longrightarrow \mathbb{Z}^+$, we need to decide if there is a subset $A' \subseteq A$ such that $\sum_{i \in A'} w(a_i) = \sum_{i \in A \setminus A'} w(a_i)$.

Given an instance of Partition, we build an instance of (UP-val) by setting $c_i = w(a_i)$ and $K_1 = K_2 = \frac{1}{2} \sum_{i \in A} w(a_i)$. Then (UP-val) is a Yes-instance iff the Partition problem is a Yes-instance.

□

Problem (UP-val) has similar structure as a knapsack problem, and also allows a pseudo-polynomial dynamic programming approach. We denote the different stages by $r = 1, \ldots, m$ and the possible right hand sides by $\lambda = 0, \ldots, val(i-1,c)+1$. We define

$$f_r(\lambda) = \min\{\sum_{i=1}^{r} c_i x_i : \sum_{i=1}^{r} c_i x_i \geq \lambda, x \in \mathbb{B}^m\}$$

and aim at finding $f_m(val(i-1,c)+1)$. To this end, we can use the following recursion:

$$f_r(\lambda) = \min\{f_{r-1}(\lambda), c_r + f_{r-1}(\lambda - c_r)\} \tag{25}$$

Using scaling techniques as for the knapsack problem [IK75], also a PTAS for (UP-val) can be achieved.

5 Interrelation between SRR and ORR

We start this section by considering interrelations between the objective values of SRR and ORR for the case of finite uncertainty sets $\mathcal{U}$. Since the solution sets $\mathcal{X}^K(c)$ and the optimal values K^* may differ between the concepts, we use an additional subindex S or O to distinguish between SRR and ORR.

By definition, $|\mathcal{X}_S^K(c)| = K \leq |\mathcal{X}|$ holds for all $c \in \mathcal{U}$, whereas $|\mathcal{X}_O^K(c)| > K$ will occur whenever there exists more than one solution with the same objective value in $P(c)$. While K_O^* is an indicator for the quality of the objective function value, K_S^* is not only related to the objective value, but also to the decision set, such that a high value of K_S^* does not necessarily indicate a solution with bad objective value. Obviously, $K_O^* \leq K_S^*$, but this result can be strengthened.

Theorem 3. *The optimal objective values K_O^* and K_S^* satisfy*

$$\frac{1}{\lfloor \frac{|\mathcal{U}|-1}{|\mathcal{U}|}|\mathcal{X}| \rfloor + 1} K_S^* \leq K_O^* \leq K_S^* \tag{26}$$

Proof. It suffices to show that $K_S^* \leq \frac{|\mathcal{U}|-1}{|\mathcal{U}|}|\mathcal{X}| + 1$. To this end, we introduce a priority function $p(x) : \mathcal{X} \longrightarrow \mathbb{N}^{|\mathcal{U}|}$ with

$$p(x) = (p_1(x), \ldots, p_{|\mathcal{U}|}(x))^T,$$

where $p_i(x)$ denotes the priority of x respective to scenario $c^i \in \mathcal{U}$ in SRR. Given $p(x)$ for some $x \in \mathcal{X}$, we can determine the maximal priority of x by

$$K^*(x) = \max_{i=1,\ldots,|\mathcal{U}|} p_i(x)$$

which gives an upper bound for K_S^*. Finding $x \in \mathcal{X}$ for which $K^*(x)$ is minimized leads to the optimal K_S^* for the SRR problem.

By definition the maximal priority of a solution x is bounded by $|\mathcal{X}|$. Since we are dealing with a finite number of scenarios, this upper bound can only be reached by at most $|\mathcal{U}|$ different solutions, i.e. $K^*(x) = |\mathcal{X}|$ can hold for at most $|\mathcal{U}|$ different solutions. With the same argument we know that for at most $|\mathcal{U}|$ different solutions we have $K^*(x) = |\mathcal{X}| - 1$ and so on. Thus, in the worst case we can iterate $K = |\mathcal{X}| - \lceil \frac{|\mathcal{X}|}{|\mathcal{U}|} \rceil$ steps without finding a K robust solution. However, in the $K+1$-th step there is at least one x with $K^*(x) = K$. By simplifying the term we get that $K_S^* \leq \frac{|\mathcal{U}|-1}{|\mathcal{U}|}|\mathcal{X}| + 1$.

□

Figure 3 illustrates this process. Here we have three scenarios and six possible solutions (A–F). The position $p_1(x)$ and the point that corresponds to the maximal priority of x are filled with the same color. Then one can see that E is the first solution for which the stopping criteria holds, and we know that the maximal possible K_S^* in this case is five.

Note that for large values of $|\mathcal{U}|$, the bound from Theorem 3 is getting close to the trivial bound $K_S^* \leq |\mathcal{X}|$; thus, our bound is particularly strong for small values of $|\mathcal{U}|$. If $|\mathcal{U}| = 2$, then Theorem 3 yields $K_S^* \leq \lfloor \frac{1}{2}|\mathcal{X}| \rfloor + 1$. We show in Appendix A that this bound is tight.

Next we assume that both solution sets $\mathcal{X}_S^{K_S^*}$ and $\mathcal{X}_O^{K_O^*}$ are given and that we have to choose a particular solution x^* as output for the SRR and the ORR

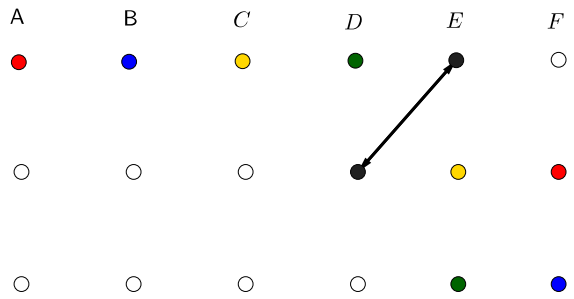

Figure 3: Example for the proof for Theorem 3.

problem, respectively. It is easy to find examples in which $\mathcal{X}_S^{K_S^*} \cap \mathcal{X}_O^{K_O^*} = \emptyset$ holds for all possible orderings of SRR. To decide which solution is put into practice, a postprocessing criterion $\mathcal{C}$ is applied, for instance one of the criteria listed at the end of Section 2.

In the final part of this section, we interpret the different objective functions $f(x, c)$ as part of a vector-valued objective function in a multi-criteria environment.

Definition 6. *We call $x \in \mathcal{X}$ **efficient** if there is no $y \in \mathcal{X}$ **dominating** x, i.e., satisfying $f(y, c) \leq f(x, c)$ for all $c \in \mathcal{U}$ and $f(y, c') < f(x, c')$ for at least one $c' \in \mathcal{U}$.*

It has been noted (see [ABV09]) that for MM and MMR at least one optimal solution is also an efficient solution. This property has also been used algorithmically, see, e.g., [CG16a, IT14]. We show that similar results hold for SRR and ORR.

Theorem 4. *Let a solution ranking be given such that efficient solutions are preferred to non-efficient solutions with the same objective value. Then, all solutions in $\mathcal{X}_S^{K^*}$ are efficient.*

Proof. Let x be in $\mathcal{X}_S^{K^*}$ and assume x is not efficient, but dominated by $y \in \mathcal{X}$. We consider the following subsets of scenarios:

$$U := \{c \in \mathcal{U} \mid f(y, c) = f(x, c)\}$$

$$U' := \{c \in \mathcal{U} \mid f(y, c) < f(x, c)\}$$

Since y dominates x, we have $U' \neq \emptyset$. By definition of $\mathcal{X}_S^K(c)$ it follows that y enters $\mathcal{X}_S^{K^*}(c)$ before x for $c \in U'$. But also for $c \in U$ we have that y enters $\mathcal{X}_S^{K^*}(c)$ before x by the assumption that efficient solutions are preferred. This contradicts the minimality of K^*; hence, x is efficient. □

If we cannot guarantee that efficient solutions are preferred, we still obtain that the solutions are weakly efficient. For ORR, a different result holds. Note that a tie breaking condition as in Theorem 4 is not required.

Proposition 1. *At least one solution in $\mathcal{X}_O^{K^*}$ is efficient.*

Proof. Assume that $x \in \mathcal{X}_O^{K^*}$ is dominated by $y \in \mathcal{X}$. We show that y is also in $\mathcal{X}_O^{K^*}$.

Let U and U' be defined as above. Then for all $c \in U'$, solution y has a strictly higher priority than x. For all $c \in U$, both x and y are contained in $\mathcal{X}_O^{K^*}(c)$. Thus, by minimality of K^*, we have that y is also included in $\mathcal{X}_O^{K^*}$. If y is dominated by some other solution $z \in \mathcal{X}$, we can repeat this argument finitely many times, until we find an optimal efficient solution. □

6 Ranking Robust Shortest Paths

We now analyze the application of ranking robustness to shortest path problems in more detail. Given a directed graph $G = (V, E)$, with edge lengths $c_e \in \mathbb{N}$, n nodes and m edges, we need to find a path from s to t with minimal length.

6.1 SRR for Shortest Path Problems

To solve SRR, we are faced with two subproblems: Determining the priority lists $\mathcal{S}(c)$, and solving the ranking problem for given priority lists. As the second problem is trivial for SRR (by checking if $\mathcal{X}^K$ is empty), we focus on the first problem here.

Finding a solution ranking is known as the K-th shortest path problem in the literature. It is known to be NP-hard (see [GJ79]), but pseudo polynomial algorithms exist. Apart from the general BST method of [HQ85], problem-specific algorithms can be found in [Yen71, EM03, ZG09].

In [ZG09], an algorithm is presented that solves the K-th shortest path problem in $\mathcal{O}(K(mn + n2loglogn))$.

6.2 ORR for Shortest Paths Problems

As before, we consider two subproblems here: The computation of $val(i, c)$, and finding a solution that is K-objective ranking robust, given the values $val(i, c)$. In contrast to SRR, the second problem is not trivial.

We first discuss the computation of $val(i, c)$ for some $c \in \mathcal{U}$, assuming that $val(i-1, c)$ has already been determined. To this end, the following IP can be used:

$$\text{(SP-val)} \qquad \min \sum_{e \in E} c_e x_e \tag{27}$$

$$\text{s.t.} \sum_{e \in E} c_e x_e \geq val(i-1, c) + 1 \tag{28}$$

$$\sum_{e \in \delta^-(v)} x_e - \sum_{e \in \delta^+(v)} x_e = b_v \qquad \forall v \in V \tag{29}$$

$$u_v - u_w + n x_e \leq n - 1 \qquad \forall e = (v, w) \in E \tag{30}$$

$$x \in \{0, 1\}^m \tag{31}$$

$$u \in \mathbb{Z}^n \tag{32}$$

where $b_s = -1$, $b_t = 1$ and $b_v = 0$ otherwise. Note that subtour elimination constraints (30) are required to find a path, as otherwise, cycles would be possible due to constraint (28).

Although the general shortest path problem is solvable in polynomial time, this does not hold for (SP-val).

Theorem 5. *Given $val(i, c)$ for some $c \in \mathcal{U}$ and $i = 1, ..., K-1$, the computation of $(SP - val)$ is NP-hard.*

Proof. The decision problem of (SP-val) is to decide if there is a simple (s, t)-path with length of at least K_1 and at most K_2. We use a reduction from the longest path problem (see [GJ79]), which is to decide if there is a simple (s, t)-path with length L or more.

Given an instance of the longest path problem, we build an instance for (SP-val) using the same graph, the same edge costs, and setting $K_1 := L$ and $K_2 = n \cdot \max_{e \in E} c_e$. Since no simple path can be longer than K_2, we find as (s, t)-of length L or longer if and only if there is an (s, t)-path with length of at least K_1 but not longer than K_2. □

We now consider the second subproblem, which is to find a *K-ORR* solution, given the values $val(i, c)$. For $K = 1$, this problem can be rewritten as

$$\min_{x \in \mathcal{X}} \left(\max_{c \in \mathcal{U}} (f(x, c) - val(1, c)) \right)$$

which is equivalent to problem *MMR* (see also Section 4).

Most combinatorial minmax regret problems are NP-hard (see [ABV09]). We show that this is also the case for *ORR* shortest paths.

Theorem 6. *Let an uncertain shortest path problem with finite uncertainty set be given. Then, the objective ranking robust shortest path problem is NP hard, even for two scenarios and if all values $val(i, c)$ are given.*

Proof. We use a reduction from the following decision problem of *MMR*: Given a graph $G = (V, E)$, is there an $s - t$ path with regret less or equal to L? This problem is known to be NP-complete already for two scenarios with integral edge lengths ([YY98]).

Given an instance of *MMR*, we construct a new graph G' where additional nodes and edges are inserted in front of node s, as described in Figure 4. The

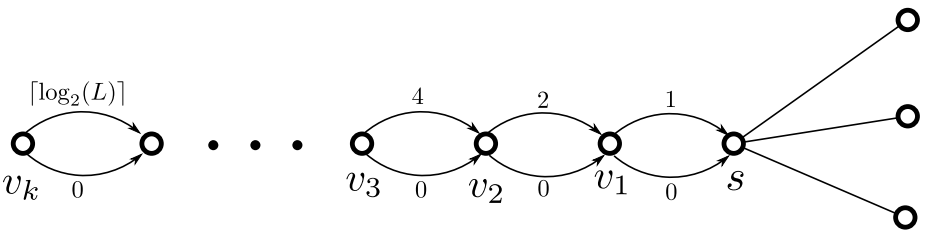

Figure 4: Graph G'.

edge costs of the new edges are constant over all scenarios. We are now looking for a path from v_k to t. Note that G' is constructed such that

$$val(K, c) = val(1, c) + K - 1$$

Accordingly, there is an $s-t$ path in G with regret at most L if and only if there is an $L+1$-objective ranking robust path in G'. □

Using the same construction, the same results also holds for spanning tree problems:

Corollary 1. *The objective ranking robust spanning tree problem is NP-hard.*

7 Computational Example

We consider the city of Kulmbach, Germany, as an example instance (see Figure 5). In the context of river flooding we focus on the river White Main which runs through Kulmbach. Due to an outdated flood protection infrastructure, which is currently being renewed, Kulmbach has been affected by recent floodings, e.g., in 2006.

Assuming that the water level increases, people need to be evacuated from endangered regions as fast as possible. To this end, we consider a network with no arc capacities such that the best evacuation route from a start to an end point can be computed by solving a shortest path problem.

Since we cannot calculate the exact degree of destruction caused by flooding or other environmental disasters, it is even more important to deal with uncertainty. Some roads might be still passable in one scenario but for another outcome it is nearly impossible to use them. Thus, we model a scenario dependent condition of roads and bridges via the arc labels of our network. Using robust optimization to determine appropriate paths leads to solutions that hedge against all given scenarios.

In this section we present an example as a proof of concept that compares our new SRR and ORR approaches with the classical concepts of minmax (MM) and minmax regret (MMR) robustness. Our intention is not to give a quantitative comparison, but to focus on qualitative solution differences.

7.1 Setup

Using OpenStreetMap (OSM) data, the underlying graph is aggregated by merging nodes that are less than 75 meters away from each other (for a detailed description of the aggregation procedure, see [Grü15]). Moreover, we removed all arcs and corresponding nodes leading to dead ends. This way, the original graph containing 4105 nodes and 59,066 arcs is reduced to 217 nodes and 690 arcs (see Figure 6).

Increasing the original, undisturbed arc lengths of the aggregated graph, we produce additional scenarios. Each arc is modified and its new weights are determined randomly using the following scheme:

$$x := uniform(l,u), \qquad y := \mathcal{N}(x, \frac{1}{2}), \qquad mod := |\mathcal{N}(y,1)|.$$

Here, $uniform(l,u)$ is the uniform distribution between $l=0$ and $u=\frac{1}{2}$ to model a set-up in which the generated scenarios deviate not that much from the original one. For stronger deviation we choose $l=\frac{1}{2}$ and $u=1$. Next, the

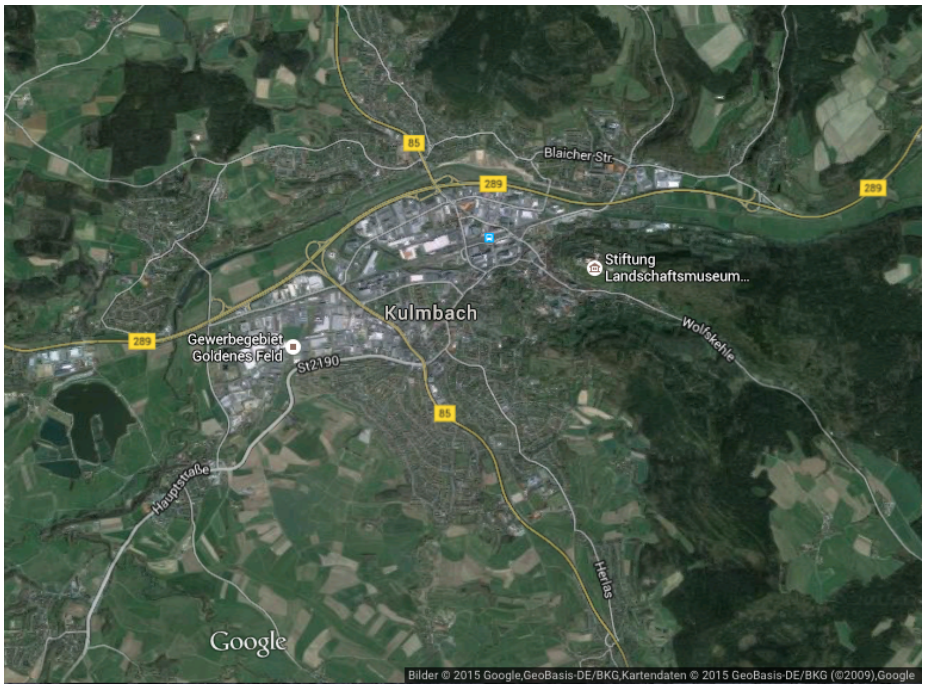

Figure 5: City of Kulmbach (image copyright 2015 Google and 2015 GeoBasis).

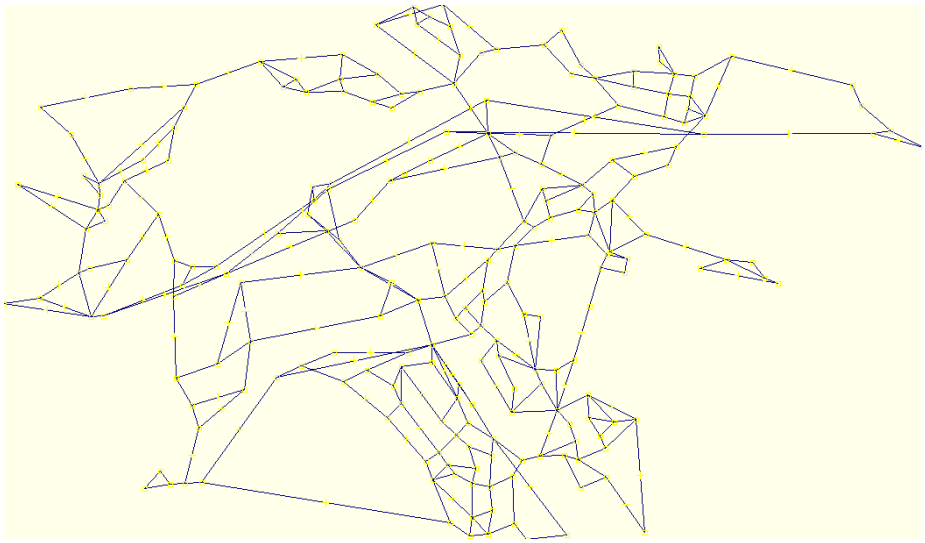

Figure 6: Aggregated OSM graph data for Kulmbach.

normal distribution $\mathcal{N}$ is applied twice to ensure that the scenarios are not too similar. When an arc length is modified, the new arc value is

$$length_{new} = length_{old}(1 + mod).$$

We created different instances and scenarios and applied *SRR*, *ORR*, *MM* and *MMR* to the resulting uncertain instance. For *MM* and *MMR* we solve the corresponding IPs using Gurobi [GO15] on Python. *SRR* is solved as described in Section 6.1, using Yen's algorithm ([Yen71]) for finding the K best shortest paths with given shortest path algorithms provided by networkx [HSS08]. For *ORR* we use Gurobi to determine solutions for the integer programs presented in Section 6.2.

All algorithms have been tested on a 64 Bit Linux compute server equipped with two Intel Xeon E5-2690 (single processor specifications: nominal speed 2.9GHz, boost up to 3.8GHz, 8 cores, 16 threads, 20MB Cache) and 192GB DDR-3 ECC RAM at 1333MHz, making use of Python 2.7.11, networkx 1.6 [HSS08], numpy 1.6.1 [SvdWV11], python-igraph 0.7.0 [CN06], and Gurobi solver 6.5.0 [GO15].

7.2 Results

We focus on an example which illustrates that our new concept is a viable alternative to *MMR* and *MM*. Extensive numerical tests will follow in a subsequent paper.

In this example, we used five scenarios and computed optimal paths with respect to *MM*, *MMR*, *ORR* and *SRR*. Since *ORR* and *SRR* lead to the same solution, we simply refer to it as *RR* in the following. The resulting three paths P_{MM}, P_{MMR} and P_{RR} are shown in Figure 7.

The path lengths with respect to the different scenarios are given in Table 3. By definition, P_{MM} has to be best in one scenario, but P_{RR} outperforms P_{MM} in all others. In only one scenario, P_{RR} is worse than both P_{MM} and P_{MMR}; in two of five scenarios it even leads to the best solution, indicating that this approach deserves further research.

	P_{RR}	P_{MM}	P_{MMR}
$c^1(P)$	6760	7306	6886
$c^2(P)$	4047	4224	3896
$c^3(P)$	8986	7904	8591
$c^4(P)$	7185	8038	7577
$c^5(P)$	7394	7675	7370

Table 3: Objective values $c^i(P)$

Tables 4–7 show how well a solution of one concept performs as a solution to a different concept. For Table 4 we computed the *ORR* objective of each of the three paths, i.e., the smallest K such that the corresponding path is K-ranking robust. In this way we obtain a quality measure for the *MMR*-solution, since the ideal minmax regret solution is $K = 1$ (see 4). The value $K = 266$ for P_{MMR} is about 1.5 times larger than the optimal $K = 188$ from P_{RR}, which indicates a potential preference for the solution obtained by the ranking approach. This

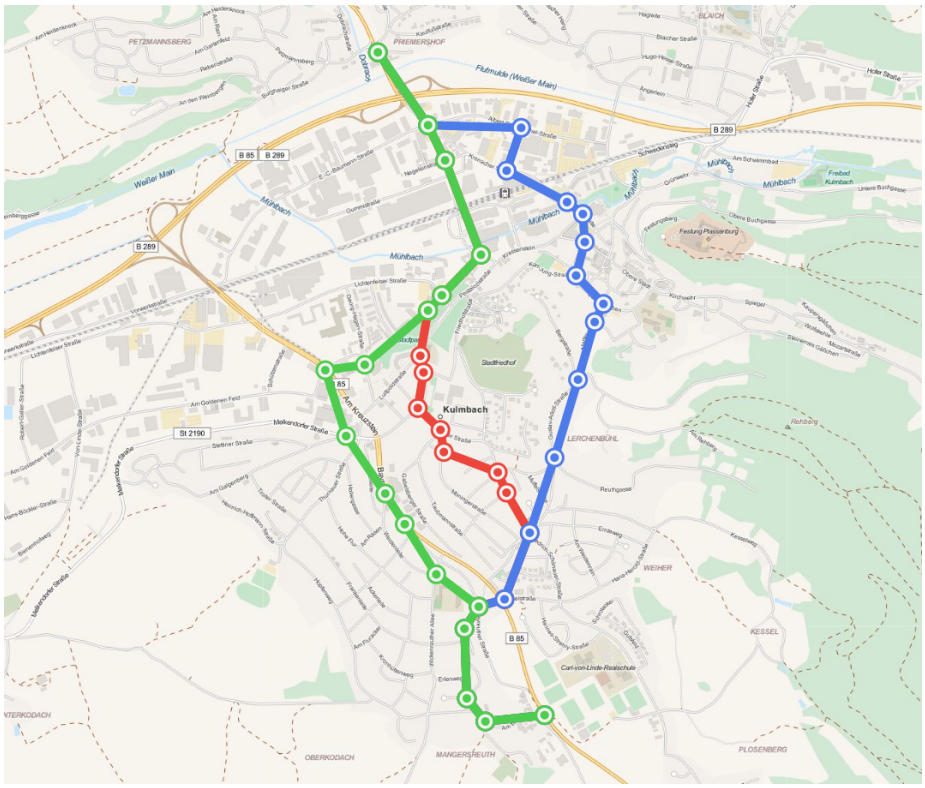

Figure 7: Shortest paths P_{RR} (green), P_{MMR} (red) and P_{MM} (blue). (© OpenStreetMap contributors `www.openstreetmap.org/copyright`)

comparison is even more in favor of RR, if we compare P_{RR} with the MM solution.

The values of Table 5 indicate in how many scenarios the solution of the various approaches are contained after the solution algorithm stopped with output K^* for SRR . By definition, this number is 5 for P_{RR}, but P_{MMR} was only contained in three, and P_{MM} even only in one K^* best priority lists.

Finally, in Tables 6 and 7 we compare the worst-case and regret objective values of all approaches. While MM performs by definition best in the worst-case, and MMR best for the regret objective, one can observe that RR presents a reasonable alternative.

Table 8 shows the computation times for each approach. While the SRR and ORR solutions have high quality, they are also more complex to compute

	Objective Value
RR	188
MM	657
MMR	266

Table 4: ORR objective values

	Number Scenarios
RR	5
MM	1
MMR	3

Table 5: Number of inclusions in $SRR - K^*$ best solutions.

	Objective Value
RR	8986
MM	8038
MMR	8591

Table 6: *MM* objective values

	Objective Value
RR	1351
MM	1754
MMR	1334

Table 7: *MMR* objective values

(especially *ORR*) and thus show a high potential for further research in the area of complexity and improving the efficiency of the underlying procedures.

Model	Time (sec.)
MM	0.23
MMR	0.33
SRR	14.93
ORR	212,747.31

Table 8: Different running times of each approach, given in seconds.

8 Conclusion

In this paper we introduced ranking robustness, a new approach to robust optimization that is based on a preference ranking of solutions. Two such ranking methods have been discussed, which are solution ranking (i.e., every feasible solution is given a unique degree of preference in every scenario) and objective ranking (i.e., solutions are ranked according to their objective value). Solution algorithms and problem complexities have been discussed for both approaches, in particular with respect to shortest path problems.

Our new approach is motivated by experience with decision makers, who tend to classify solutions according to a coarser concept of quality than their precise objective value.

As a proof of concept our approach has been applied to a real-world shortest path instance, where we observed promising differences to other robust solutions, motivating further research into ranking robustness.

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*, 197:427–438, 2009.

[BBC11] D. Bertsimas, D. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[BBI14] F. Baumann, C. Buchheim, and A. Ilyina. A Lagrangean decomposition approach for robust combinatorial optimization. Technical report, Optimization Online, 7 2014.

[BMSW13] Joachim M Buhmann, Matús Mihalák, Rastislav Srámek, and Peter Widmayer. Robust optimization in the presence of uncertainty.

In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 505–514. ACM, 2013.

[BTGN09] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, Princeton and Oxford, 2009.

[BTN98] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[CG16a] A. Chassein and M. Goerigk. A bicriteria approach to robust optimization. *Computers & Operations Research*, 66:181 – 189, 2016.

[CG16b] A. Chassein and M. Goerigk. Performance analysis in robust optimization. In E. Grigoroudis M. Doumpos, C. Zopounidis, editor, *Robustness Analysis in Decision Aiding, Optimization, and Analytics*, International Series in Operation Research & Management Science. Springer, 2016. To appear.

[CN06] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[Ehr06] Matthias Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2006.

[EM03] M.M.B. Pascoal E.Q.V. Martins. A new implementation of Yen's ranking loopless paths algorithm. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(2):121–134, 2003.

[GJ79] M R Garey and D S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman & Co., San Francisco, 1979.

[GO15] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015.

[Grü15] B. Grün. *Dissertation tba*. PhD thesis, Technische Universität Kaiserslautern, 2015.

[GS15] 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 *Lecture Notes in Computer Science*, page 0. Springer Berlin / Heidelberg, 2015.

[GYdH15] Bram L Gorissen, İhsan Yanıkoğlu, and Dick den Hertog. A practical guide to robust optimization. *Omega*, 53:124–137, 2015.

[HQ85] H.W. Hamacher and M. Queyranne. K best solutions to combinatorial optimization problems. *Annals of Operations Research*, 4:123–143, 1985.

[HSS08] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[HT01] H. W. Hamacher and S. A. Tjandra. Mathematical modelling of evacuation problems: a state of the art. In *Pedestrian and Evacuation Dynamics*, pages 227–266. Springer, Berlin, 2001.

[IK75] Oscar H Ibarra and Chul E Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM)*, 22(4):463–468, 1975.

[IT14] Dan A. Iancu and Nikolaos Trichakis. Pareto efficiency in robust optimization. *Management Science*, 60(1):130–147, 2014.

[MG04] R Montemanni and L.M. Gambardella. An exact algorithm for the robust shortest path problem with interval data. *Computers & Operations Research*, 31(10):1667–1680, 2004.

[Mie14] K. Miettinen. Survey of methods to visualize alternatives in multiple criteria decision making problems. *OR Spectrum*, 36(1):3–37, 2014.

[SvdWV11] S. Chris Colbert Stéfan van der Walt and Gaël Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13:22–30, 2011.

[Yen71] J. Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17:712–716, 1971.

[YY98] Gang Yu and Jian Yang. On the Robust Shortest Path Problem. *Computers & Operations Research*, 25(6):457–468, 1998.

[ZG09] M. Lewenstein Z. Gotthilf. Improved algorithms for the k simple shortest paths and the replacement paths problems. *Inf. Process. Lett.*, 109:352–355, 2009.

A Tightness of Bound in Theorem 3

We show that the bound

$$\frac{1}{\lfloor \frac{|\mathcal{U}|-1}{|\mathcal{U}|}|\mathcal{X}| \rfloor + 1} K_S^* \leq K_O^*$$

is tight whenever $|\mathcal{U}| < |\mathcal{X}|$. For $|\mathcal{U}| \geq |\mathcal{X}|$ the trivial bound $K_S^* \leq |\mathcal{X}|$ holds.

Let $\mathcal{X} = \{x_1, \ldots, x_N\}$, $\mathcal{U} = \{c^1, \ldots, c^M\}$ and $N - 1 = s(M-1) + r$. For $i = 1, ..., N$ we set $f(x_i, c^1) = 1$. Furthermore, we set

$$f(x_1, c^2) = f(x_2, c^3) = \cdots = f(x_{M-1}, c^M) = N,$$
$$f(x_M, c^2) = f(x_{M+1}, c^3) = \cdots = f(x_{2M-2}, c^M) = N - 1,$$

	x_1	$\cdots$	x_{M-1}	x_M	$\cdots$	$x_{2(M-1)}$	$\cdots$	$x_{s(M-1)+1}$	$\cdots$	x_{N-1}	x_N
c^1	1	$\cdots$	1	1	$\cdots$	1	$\cdots$	1	$\cdots$	1	1
c^2								$N-s$	$\cdots$	1	1
$\vdots$									$\ddots$		
c^{r+1}	$\mathbf{1}+(N-1)\cdot Id$			$\mathbf{1}+(N-2)\cdot Id$			$\cdots$	1	$\cdots$	$N-s$	1
c^{r+2}								1	$\cdots$	1	1
$\vdots$									$\ddots$		
c^M								1	$\cdots$	1	1

Table 9: Instance for which the inequality of Theorem 3 is tight.

and so on, until we reach x_N. $f(x_N, c^i)$ for $i = 1, ..., M$ and all remaining objective values are set to one (see Table 9).

For this instance, the ORR problem is solved with $K_O^* = 1$ and $\mathcal{X}_O^{K_O^*} = \{x_N\}$. We can solve the SRR problem by setting $p_1(x_i) = i$ for $i = 1, ..., N$ which is possible since all objective values are the same. Then, K_S^* is determined by finding x_i with $p_1(x_i) \geq K^*(x_i)$. By construction we obtain this for $i = \lfloor \frac{M-1}{M} N \rfloor + 1$. This way we can always find an instance for which the ratio of Theorem 3 holds with equality.

Note that this example only works since the ranking of solutions with equal objective value can be done arbitrarily. If there is an ordering rule, better bounds might be obtained.