Approximation Algorithms for Min-max-min Robust Optimization and K-Adaptability under Objective Uncertainty

Jannis Kurtz*[1]

[1]Amsterdam Business School, University of Amsterdam, 1018 TV Amsterdam, Netherlands

Abstract

In this work we investigate the min-max-min robust optimization problem and the k-adaptability robust optimization problem for binary problems with uncertain costs. The idea of the first approach is to calculate a set of k feasible solutions which are worst-case optimal if in each possible scenario the best of the k solutions is implemented. It is known that the min-max-min robust problem can be solved efficiently if k is at least the dimension of the problem, while it is theoretically and computationally hard if k is small. However, nothing is known about the intermediate case, i.e. k lies between one and the dimension of the problem. We approach this open question and present an approximation algorithm which achieves good problem-specific approximation guarantees for the cases where k is close to or where k is a fraction of the dimension. The derived bounds can be used to show that the min-max-min robust problem is solvable in oracle-polynomial time under certain conditions even if k is smaller than the dimension. We extend the previous results to the robust k-adaptability problem. As a consequence we can provide bounds on the number of necessary second-stage policies to approximate the exact two-stage robust problem. We derive an approximation algorithm for the k-adaptability problem which has similar guarantees as for the min-max-min problem. Finally, we test both algorithms on knapsack and shortest path problems and related two-stage variants. The experiments show that both algorithms calculate solutions with relatively small optimality gap in seconds.

Robust Optimization Min-max-min K-adaptability Approximation Algorithm

1 Introduction

Integer optimization problems nowadays emerge in many industries as production, health care, disaster management or transportation, just to name a few. The latter problems are tackled by companies, non-profit organizations or governmental institutions and providing good solutions is a highly relevant topic in our society. Usually solving optimization problems in practice involves uncertainties which have to be incorporated into the optimization model. Typical examples are uncertain traffic situations, demands or failures of a network. The optimization literature provides several ways to model uncertainties, e.g. stochastic optimization [12], robust optimization [4] or distributionally robust optimization [44, 29].

In *robust optimization* we assume that all possible realizations of the uncertain parameters are contained in a given uncertainty set and the aim is to find a solution which is optimal in the

*j.kurtz@uva.nl

worst-case and feasible for all possible scenarios in the uncertainty set. The robust optimization approach was intensively studied for convex and discrete uncertainty sets; see e.g. [6, 7, 11, 10, 34, 2, 17]. Despite its success it can be too conservative since the calculated robust solution has to hedge against all scenarios in the given uncertainty set, which can lead to bad performances in the mean scenario. To overcome this problem several new robust models have been introduced; see e.g. [5, 24, 36, 1].

In this work we study the *min-max-min robust optimization problem*, which was first introduced in [15] to overcome the conservativeness of the classical robust approach. We consider deterministic optimization problems

$$\min_{x \in X} c^\top x \tag{P}$$

where $X \subseteq \{0,1\}^n$ is the set of incidence vectors of all feasible solutions and $c \in \mathbb{R}^n_+$ is an uncertain cost-vector which we assume is contained in a given convex uncertainty set $U \subseteq \mathbb{R}^n_+$. Note that most of the combinatorial problems can be modeled as binary problems and that the restriction to binary variables is theoretically no limitation as general integer variables can be modeled by binary variables using binary encoding.

Similar to the idea of k-adaptability ([8, 32, 43]) the main idea of the min-max-min robust optimization problem is to hedge against the uncertainty in the cost-vector in a robust manner, i.e. considering the worst-case costs over all scenarios in U, while providing more flexibility compared to the classical robust approach since multiple solutions are calculated and can be used to react to the emerging uncertain scenarios. In contrast to the k-adaptability problem the min-max-min robust approach does not consider two-stage structures and is therefore tailored for binary problems where no second-stage decisions exist, i.e. where a set of complete solutions has to be prepared in advance. This can be inevitable in many applications regarding the construction of transportation plans, e.g. evacuation plans for buildings, airports or cities [18, 39, 37, 38] or route plans for robot systems [40]. Note that the restriction to objective uncertainty is still an interesting case since despite many combinatorial problems, also real-world problems appear which can be modeled by objective uncertainty (see e.g. [41]). Furthermore recent works on two-stage robust optimization indicate that it is possible to use Lagrangian relaxation to shift uncertain constraints into the objective function; see [42, 35].

Since 2015 the min-max-min robust approach was studied for several uncertainty sets and combinatorial problems. It was studied for convex uncertainty sets in [15, 14] and for discrete uncertainty sets in [16]. Regarding its complexity and solvability the min-max-min robust problem is a very interesting problem due to the unusual connection between its problem parameters and its complexity. A reasonable assumption would be that the problem gets harder to solve with increasing k. However this is not true from a theoretical as well as from a computational point of view. While for discrete uncertainty sets it is weakly or strongly NP-hard for most of the classical combinatorial problems [16], in the case of convex uncertainty the problem can be solved in polynomial time if $k \geq n$ and if (P) can be solved in polynomial time [15]. On the other hand it is NP-hard for each fixed $k \in \mathbb{N}$ even if U is a polyhedron. The authors in [15] present an efficient algorithm for the case $k \geq n$ and a fast heuristic for each fixed $k \in \mathbb{N}$. Later in [19] it was shown that the problem can be solved in polynomial time for several combinatorial problems if U is a convex budgeted uncertainty set and $k = 2$. In [20] faster exact and heuristic algorithms for the same uncertainty set were presented. For the discrete budgeted uncertainty set the authors in [28] derive exact and fast heuristic algorithms and show that the problem is weakly or strongly NP-hard for most of the classical combinatorial problems. For binary uncertainty sets defined by linear constraints it was shown in [21] that the min-max-min robust problem is Σ^p_2-complete. Recently an efficient exact algorithm based on smart enumeration was derived in [3] for problems where X does not contain too many good solutions. In [23] the min-max-min robust problem

was applied to the vehicle routing problem where a set of k possible routes has to be prepared in advance which are robust to uncertain traffic scenarios. The idea of the min-max-min robust approach was also applied to the regret robust approach in [22].

However, no efficient algorithms are known which solve the min-max-min problem for any $k \in \mathbb{N}$ efficiently with a certain approximation guarantee. Furthermore nothing is known about the complexity of the problem if k has intermediate size.

As mentioned above the min-max-min problem has a similar structure as the *k-adaptability problem* which was introduced in [8] to approximate two-stage robust optimization problems with integer second-stage variables (recently a short note on the incorrectness of the continuity assumption made in the latter paper appeared [33]). The idea is to calculate k second-stage policies already in the first stage and choose the best of it after the scenario is revealed. This idea provides a heuristic solution to the exact two-stage robust problem and no approximation guarantees were known so far. Due to the similar structure, algorithmic ideas from the k-adaptability literature can also be applied to the min-max-min problem. In [32] a mixed-integer programming formulation was derived to solve the k-adaptability problem. Later in [43] the authors present an algorithm based on a branch & bound scheme which iteratively constructs partitions of scenarios. In [25] a logic-based benders decomposition approach is used where the main problem calculates a first-stage solution while a subproblem is used to derive cuts on the objective value. Interestingly the subproblem is a classical min-max-min problem and the authors use methods from the min-max-min literature to solve it. Finally in [9] geometric properties of the uncertainty set are used to derive approximation guarantees for the k-adaptability problem with right-hand side uncertainty.

From a theoretical perspective there is not much known about the approximation guarantees the k-adaptability approach provides for the two-stage robust problem. The only result into that direction was proved in [32] where it is shown that the k-adaptability problem provides an optimal solution of the exact two-stage problem if the number of second-stage policies is at least the dimension of the problem. We will close this gap in this work, providing better bounds on k which lead to a certain problem-specific approximation guarantee. Furthermore we provide the first approximation algorithms with a provable approximation guarantee for the k-adaptability problem.

Contributions

- We provide efficient algorithms to calculate solutions with provable additive and multiplicative approximation guarantees for the cases where a) k is smaller but close to n and b) k is a fraction of n. The derived guarantees hold for a wide class of binary problems and involve a problem-specific parameter.
- We use the approximation guarantees to derive ranges for parameter k for which the algorithm provides a certain approximation guarantee.
- We show for the first time that under certain assumptions the min-max-min robust problem remains oracle-polynomial solvable if $k = n - l$ and l is a fixed parameter.
- We extend the derived approximation guarantees to the k-adaptability approach and show that they can be used to calculate better bounds for the number of second-stage policies k which are necessary to achieve a certain approximation guarantee for the exact two-stage robust problem.
- We derive an efficient approximation algorithm which calculates solutions for the k-adaptability problem with a certain approximation guarantee.

- We perform experiments to test both approximation algorithms on knapsack and shortest path instances and on a generic k-adaptability problem and a two-stage network construction shortest path problem.

The paper is organized as follows. In Section 2 we provide preliminary results on min-max-min robustness and k-adaptability. In Section 3 we provide the approximation algorithm and derive additive and multiplicative approximation guarantees depending on n and k. In Section 4 we extend the derived results to the k-adaptability approach and provide better bounds on the parameter k to achieve a certain approximation guarantee. Afterwards we present an approximation algorithm and prove its approximation guarantee. Finally in Section 5 we show the results of our computational experiments and give a conclusion in Section 6.

2 Preliminaries

2.1 Notation

We define $[k] := \{1, \dots, k\}$ for $k \in \mathbb{N}$ and $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n : x \geq 0\}$. We denote by $\|x\| := \sqrt{\sum_{i\in[n]} x_i^2}$ the euclidean norm and by $\|x\|_\infty := \max_{i\in[n]} |x_i|$ the maximum norm. The convex hull of a finite set $S = \{s^1, \dots, s^m\}$ is denoted by

$$\operatorname{conv}(S) = \left\{ s = \sum_{i\in[m]} \lambda_i s^i : \lambda_i \geq 0 \ \forall i \in [m],\ \sum_{i\in[m]} \lambda_i = 1 \right\}.$$

The vector of all ones is denoted by $\mathbf{1}$ and the i-th unit vector by e_i.

2.2 Min-max-min Robust Optimization

Formally the min-max-min robust optimization problem is defined as

$$\min_{x^{(1)},\dots,x^{(k)}\in X} \max_{c\in U} \min_{i=1,\dots,k} c^\top x^{(i)} \tag{M^3(k)}$$

where $X \subseteq \{0,1\}^n$, $U \subseteq \mathbb{R}^n_+$ is a convex uncertainty set and $k \in \mathbb{N}$ is a given parameter controlling the conservativeness of the problem. In [15] the authors study Problem (M^3(k)) for convex uncertainty sets U and show, by using lagrangian relaxation, that Problem (M^3(k)) for any $k \in \mathbb{N}$ is equivalent to problem

$$\min_{x\in X(k)} \max_{c\in U} c^\top x \tag{1}$$

where $X(k)$ is the set of all convex combinations derived by at most k solutions in X, i.e.

$$X(k) := \left\{ x \in \mathbb{R}^n :\ x = \sum_{i\in[k]} \lambda_i x^{(i)},\ x^{(i)} \in X,\ \lambda \in \mathbb{R}^k_+,\ \sum_{i\in[k]} \lambda_i = 1 \right\}.$$

By the theorem of Carathéodory it follows that each point in $\operatorname{conv}(X)$ can be described by a convex combination of at most $n+1$ points in X, therefore it holds $X(k) = \operatorname{conv}(X)$ for all $k \geq n+1$. Since for any given point $x \in \operatorname{conv}(X)$ and $\mu \in \mathbb{R}_+$ we have

$$\max_{c\in U} c^\top(\mu x) = \mu \max_{c\in U} c^\top x$$

an optimal solution is always attained on the boundary of conv (X) if $k \geq n+1$, i.e. can be described by a convex combination of at most n solutions in X. It follows that for each $k \geq n$ Problem $(\mathrm{M}^3(k))$ is equivalent to the problem

$$\min_{x \in \mathrm{conv}(X)} \max_{c \in U} c^\top x. \tag{2}$$

From the latter result we obtain the following chain of optimal values, where we denote by $\mathrm{opt}(k)$ the optimal value of $(\mathrm{M}^3(k))$ with k solutions:

$$\mathrm{opt}(1) \geq \mathrm{opt}(2) \geq \ldots \geq \mathrm{opt}(n) = \mathrm{opt}(n+1) = \ldots.$$

Note that $\mathrm{opt}(1)$ is equal to the optimal value of the classical robust problem.

For an optimal solution x^* of Problem (2) the corresponding optimal solution $x^{(1)}, \ldots, x^{(n)}$ of Problem $(\mathrm{M}^3(n))$ can be calculated in polynomial time, if we can linearly optimize over X in polynomial time; see [30, 15] for more details.

Problem (2) is a convex problem, since the objective function $f(x) := \max_{c \in U} c^\top x$ is convex and conv (X) is a convex set. Unfortunately for many classical combinatorial problems no outer-description of polynomial size for conv (X) is known. Nevertheless the authors in [15] prove that Problem (2) and therefore the min-max-min robust problem can be solved in polynomial time if the underlying deterministic problem (P) can be solved in polynomial time. More precisely the theorem states that, if we can linearly maximize over U in polynomial time and if we can linearly minimize over X in polynomial time, then we can solve the min-max-min robust problem in polynomial time. However the proof in [15] is not constructive, i.e. no implementable algorithm with a polynomial runtime guarantee is presented. Instead the authors present a column-generation algorithm to solve Problem $(\mathrm{M}^3(k))$ for $k \geq n$ where iteratively the deterministic problem (P) and an adversary problem over U is solved. They show that this algorithm is very efficient on random instances of the knapsack problem and the shortest path problem. Furthermore the same algorithm was used successfully in [23] for the min-max-min version of the capacitated vehicle routing problem. We will adapt this algorithm to find good approximate solutions and calculate strong lower bounds for Problem $(\mathrm{M}^3(k))$ for any k.

On the other hand it is shown in [15] that Problem $(\mathrm{M}^3(k))$ with an uncertain constant is NP-hard for any fixed $k \in \mathbb{N}$, even if U is a polyhedron given by an inner description and $X = \{0,1\}^n$. This result fits to the computational experience which was made in other publications, where it turns out that Problem $(\mathrm{M}^3(k))$ is very hard to solve for small $k \in \mathbb{N}$; see [3, 20]. Nevertheless to tackle the problem even for small k in [15] the authors present an heuristic algorithm which is based on the column-generation algorithm mentioned above. The idea is to solve Problem $(\mathrm{M}^3(k))$ for $k = n$ with this algorithm and afterwards select k of the calculated solutions with largest induced weights, given by the optimal convex combination of Problem (2). It is shown computationally that this heuristic calculates solutions which are very close to the optimal value of Problem $(\mathrm{M}^3(k))$. We will present a theoretical understanding of this behavior in Section 3 for the first time and provide approximation bounds.

2.3 Two-stage robust optimization and k-adaptability

The k-adaptability problem was introduced with the goal to approximate two-stage robust problems with second-stage integer variables in [8]. We define the two-stage robust problem with objective uncertainty as

$$\min_{x \in X} \max_{\xi \in U} \min_{y \in Y(x)} d^\top x + \xi^\top y \tag{2RO}$$

where $X \subseteq \mathbb{R}^m$, $Y(x) \subseteq \{0,1\}^n$ for all $x \in X$ and $U \subseteq \mathbb{R}^n_+$ is a convex uncertainty set. The variables x are the first-stage decisions which have to be taken before the uncertainty reveals.

The variables y are second-stage decisions which can be taken after the uncertain parameters ξ are known. Note that the feasible set $Y(x)$ depends on the taken first-stage decision. While the following results hold for the more general case, in the linear case often the second-stage feasibility set is given as

$$Y(x) = \{y \in Y \subseteq \{0,1\}^n \mid Tx + Wy \leq h\}$$

for matrices T, W and vector h of appropriate size. The vector $d \in \mathbb{R}^m_+$ contains the given first-stage costs which we assume to be not uncertain, although all the following results are still valid otherwise.

The k-adaptable problem was already studied in [8, 32, 43, 25]. The idea is to calculate a set of k second-stage solutions $y^1, \ldots, y^k \in Y(x)$ already in the first-stage and select the best one for each scenario $\xi \in U$. This problem can be formulated as

$$\min_{\substack{x \in X, \\ y^1,\ldots,y^k \in Y(x)}} \max_{\xi \in U} \min_{i=1,\ldots,k} d^\top x + \xi^\top y^i. \tag{k-adapt}$$

Note that while the k-adaptability problem is complexity-wise harder than the min-max-min problem ($\mathrm{M}^3(k)$), both problems have a quite similar structure. Namely, if we fix a first-stage solution x, then (k-adapt) becomes a classical min-max-min robust problem ($\mathrm{M}^3(k)$).

We can reformulate (k-adapt) as

$$\min_{\substack{x \in X, \\ y^1,\ldots,y^k \in Y(x)}} \max_{\xi \in U} \min_{i=1,\ldots,k} d^\top x + \xi^\top y^i = \min_{\substack{x \in X, \\ y^1,\ldots,y^k \in Y(x)}} \max_{\xi \in U} \min_{\substack{\lambda \geq 0, \\ \sum_i \lambda_i = 1}} d^\top x + \sum_{i=1}^k \lambda_i \xi^\top y^i.$$

Since the feasible regions of the inner max and the inner min are both convex, and the objective function is linear in both variables, we can swap the maximum and the inner minimum, which leads to the formulation

$$\min_{\substack{x \in X,\ \lambda \geq 0 \\ y = \sum_{i \in [k]} \lambda_i y^i \\ \sum_{i \in [k]} \lambda_i = 1 \\ y^1,\ldots,y^k \in Y(x)}} \max_{\xi \in U} d^\top x + \xi^\top y.$$

Note that each feasible solution y of the latter problem is a convex combination of k feasible second-stage policies. Since each point in the convex hull can be described by a convex combination of at most $n+1$ points (by Theorem of Carathéodory), it follows from the last formulation that (k-adapt) attains the exact optimal value of (2RO) for each $k \geq n+1$ which was proved in [32]. In the following we denote by $\mathrm{opt}(2RO)$ the optimal value of the two-stage robust problem (2RO). Furthermore we denote by $\mathrm{adapt}(k)$ the optimal value of (k-adapt) with k second-stage policies. Note that by the previous analysis it holds $\mathrm{opt}(2RO) = \mathrm{adapt}(n+1)$. Furthermore we obtain the following chain of optimal values:

$$\mathrm{adapt}(1) \geq \mathrm{adapt}(2) \geq \ldots \geq \mathrm{adapt}(n) \geq \mathrm{opt}(2RO) = \mathrm{adapt}(n+1) = \mathrm{adapt}(n+2) = \ldots$$

where $\mathrm{adapt}(1)$ is equal to the optimal value of the classical robust problem.

3 Approximation algorithm for min-max-min robust optimization

The min-max-min robust problem is known to be easy to solve if $k \geq n$, both theoretically and practically (see [15]), while it is NP-hard for any fixed $k \in \mathbb{N}$ even if U is a polyhedron

and $X = \{0,1\}^n$. Moreover recent results indicate that it is computationally very hard to solve ($\mathrm{M}^3(k)$) exactly for general convex uncertainty sets U even if k is fixed and small, e.g. $k \in \{2,3,4\}$; see [15, 20, 3]. However nothing is known for the intermediate setting, i.e. if k is not small but smaller than n. Furthermore no algorithms with approximation guarantees are known for the problem.

In this section we approach this gap by answering the following research questions.

RQ1. Can we derive an algorithm with general additive and multiplicative approximation guarantees for Problem ($\mathrm{M}^3(k)$) for any number of solutions k?

RQ2. For which range of k is a certain approximation guarantee valid?

RQ3. What is the theoretical complexity of Problem ($\mathrm{M}^3(k)$) for an intermediate number of solutions k?

We first have to define what we mean by "intermediate size of k". To this end we consider two cases where a) $k = n - l$ for a fixed $l \in [n-1]$ and b) $k = qn$ for a fixed $q \in (0,1)$. Case a) can be interpreted as "k is close to n" while case b) means "k is a fraction of n" where the parameter q controls the distance to 1 and to n.

In Algorithm 1 we present an efficient algorithm which calculates feasible solutions for ($\mathrm{M}^3(k)$). The algorithm was already studied computationally in [15]. In this work we study the approximation performance of Algorithm 1. Note that Step (1) can be solved theoretically in oracle-polynomial time in the input values; see [15]. This step can be implemented by using any algorithm for the case $k \geq n$, e.g. the oracle-based column-generation algorithm presented in [15]. In Step (2) we then calculate the optimal coefficients of the convex combination of the calculated n solutions and afterwards select only the k solutions with largest coefficients. The maximum expression in the problem of Step (2) can be dualized for classical uncertainty sets (e.g. polyhedra or ellipsoids). Hence this step involves solving a continuous convex optimization problem which can be done in polynomial time for polyhedral or ellipsoidal uncertainty sets. In [15] it was shown that the algorithm is computationally very efficient and provides solutions which are often close to optimal. However no theoretical understanding of this behavior is known. In this section we will, for the first time, derive additive and multiplicative problem-specific approximation guarantees for Algorithm 1. Furthermore we will show how to calculate ranges for k for which a given approximation guarantee holds. The derived results can be used to show that ($\mathrm{M}^3(k)$) is actually oracle-polynomial solvable under certain conditions even if $k < n$.

Remark 1 (Variant of Algorithm 1)**.** *Note that we can add a sparsity constraint to the problem in Step 2 which ensures that at most k of the optimal λ_i values are non-zero. More precisely we can solve the following problem in Step 2 in Algorithm 1:*

$$
\begin{aligned}
\min \ & \max_{c \in U} \ c^\top x \\
s.t. \quad & \sum_{i=1}^{n} \lambda_i = 1 \\
& x = \sum_{i=1}^{n} \lambda_i x^i \\
& \lambda_i \leq u_i \quad i = 1, \ldots, n \\
& \sum_{i=1}^{n} u_i \leq k \\
& \lambda \in \mathbb{R}^n_+, u \in \{0,1\}^n
\end{aligned}
$$

Algorithm 1 Approximation Algorithm for ($\mathrm{M}^3(k)$)

Input: $n \in \mathbb{N}$, $k \in [n]$, convex set $U \subset \mathbb{R}^n$, $X \subseteq \{0,1\}^n$

1: calculate an optimal solution $x^1, \ldots, x^n$ of ($\mathrm{M}^3(n)$)

2: calculate an optimal solution λ^* of

$$\begin{aligned} \min_{\lambda \geq 0} \ & \max_{c \in U} \ c^\top x \\ s.t. \quad & \sum_{i=1}^{n} \lambda_i = 1 \\ & x = \sum_{i=1}^{n} \lambda_i x^i \end{aligned}$$

3: sort the λ^* values in decreasing order

$$\lambda^*_{i1} \geq \lambda^*_{i2} \geq \ldots \geq \lambda^*_{in}$$

4: **Return:** $x^{i1}, \ldots, x^{ik}$

Since the latter problem selects the best k of the n solutions involved in the convex combination, the solution returned by Algorithm 1 when using the latter problem has an objective value at most as large as the original solution of Algorithm 1. Hence all the approximation results shown in this paper also hold for this variant of the algorithm. On the other hand the latter problem is computationally harder since it involves a sparsity constraint. In our experiments in Section 5 we show that indeed the solutions of the variant have better objective values coming along with a slightly larger computation time. We will denote the variant of the algorithm as Algorithm 1 (Variant).

In the following we denote the exact optimal value of Problem ($\mathrm{M}^3(k)$) by $\mathrm{opt}(k)$ and the objective value of the solution returned by Algorithm 1 as $\mathrm{approx}(k)$. We say Algorithm 1 has an *additive approximation guarantee* of $a : \mathbb{N} \to \mathbb{R}_+$ if

$$\mathrm{approx}(k) \leq \mathrm{opt}(k) + a(n)$$

for all $n \in \mathbb{N}$. We say Algorithm 1 has an *multiplicative approximation guarantee* of $a : \mathbb{N} \to \mathbb{R}_+$ if

$$\mathrm{approx}(k) \leq (1 + a(n)) \, \mathrm{opt}(k)$$

for all $n \in \mathbb{N}$. Note that a can also be a constant function, e.g. $a(n) \equiv \varepsilon$ for any $\varepsilon > 0$ and we say the algorithm has a *constant approximation guarantee* in this case.

We assume that we have an oracle which returns an optimal solution of the deterministic problem (P) for each objective cost vector $c \in U$ in constant time. If we speak in the following of an oracle-polynomial algorithm this means that the calculations for the deterministic problem are assumed to be constant.

We assume that $\|c\|_\infty \leq M_\infty$ and $\|c\|_\infty \geq m_\infty > 0$ for all $c \in U$. While assuming an upper bound on U is more natural, the assumption on the lower bound seems restricting. However all results in Section 3.1 can be derived without assuming a lower bound m_∞ leading to non-substantially worse guarantees. For the multiplicative bounds in Section 3.2 the assumption is needed since for non-positive objective values the definition of multiplicative approximation guarantees is not well-defined. From a practical point of view the assumption is reasonable since

for many combinatorial problems the objective costs can be assumed to be strictly positive. Note that considering a maximum-norm bound for any $n \in \mathbb{N}$ is less restrictive than using the euclidean-norm since the latter grows with increasing n even if the entries of the scenarios remain of the same size. In the following we denote the number of non-zero entries of $x \in X$ by $\|x\|_0$, i.e.

$$\|x\|_0 := |\{i \in [n] : x_i = 1\}|.$$

We furthermore assume that for a given problem class there exist functions $\underline{p}, \bar{p} : \mathbb{N} \to \mathbb{R}_+$ such that for each instance X of the problem class of dimension n it holds $\underline{p}(n) \leq \|x\|_0 \leq \bar{p}(n)$ for all $x \in X$. Note that while $\underline{p}(n) \equiv 1$ and $\bar{p}(n) = n$ are always valid functions it is possible to derive problem-specific tighter functions as shown in the following proposition. The problem specific functions $\underline{p}, \bar{p}$ will later appear in the derived approximation guarantees.

Proposition 2. *In the following we present functions $\underline{p}, \bar{p}$ for a list of combinatorial problems.*

a) For the spanning tree problem defined on a graph $G = (V, E)$ for each solution x we have $\|x\|_0 = |V| - 1$, i.e. $\underline{p}(n) = 1$ and $\bar{p}(n) \leq n$. If G is a complete graph, then the number of edges is $|V|(|V| - 1)$, i.e. for each given n the number of vertices $|V|$ is uniquely defined and we have $\underline{p}(n) = \bar{p}(n) \leq \sqrt{n}$.

b) For the matching problem defined on a graph $G = (V, E)$ for each solution x we have $\|x\|_0 \leq \frac{1}{2}|V|$, i.e. $\underline{p}(n) = 1$ and $\bar{p}(n) \leq n$. If G is a complete graph (which is the case for the assignment problem), then the number of edges is at most $|V|^2$, i.e. we have $\underline{p}(n) = \bar{p}(n) \leq \sqrt{n}$.

c) For each cardinality constrained problem, i.e. $X_c = \{x \in X \mid \sum_i^n x_i = \bar{p}\}$ for a given fixed $\bar{p} \in [n]$, we have $\underline{p}(n) = \bar{p}(n) = \bar{p}$. One popular example from robust combinatorial optimization is the p-selection problem where $X = \{0, 1\}^n$.

d) For the traveling salesmen problem (TSP) defined on a complete graph $G = (V, E)$ for each solution x we have $\|x\|_0 = |V|$, i.e. $\underline{p}(n) = \bar{p}(n) \leq \sqrt{n}$.

e) For the vehicle routing problem (VRP) defined on a complete graph $G = (V, E)$ with m customers and one depot for each solution x we have $\|x\|_0 \leq 2m$ (in case each vehicle visits exactly one customer) and $m+1 \leq \|x\|_0$ (in case only one vehicle visits all customers). Since the graph has $m(m + 1)$ edges we have $\bar{c}\sqrt{n} \leq \underline{p}(n) \leq \bar{p}(n) \leq \bar{C}\sqrt{n}$ for constants $\bar{c}, \bar{C} \geq 0$.

3.1 Additive approximation guarantees

We first prove the following general lemma which generalizes the result used in the proof of Theorem 6 in [15].

Lemma 3. *Assume $s, k \in [n]$ where $s < k$, then it holds*

$$\text{opt}(s) - \text{opt}(k) \leq M(n)\frac{k - s}{s + 1}.$$

where $M(n) := M_\infty \bar{p}(n) - m_\infty \underline{p}(n)$.

Proof. Let $x^*(k)$ be an optimal solution of Problem (1) with parameter k. Then by the results in [15] we have

$$\text{opt}(k) = \max_{c \in U} c^\top x^*(k)$$

and there exists a convex combination $x^*(k) = \sum_{i\in[k]} \lambda_i x^i$ where $x^i \in X$ and $\lambda \in \mathbb{R}^k_+$ with $\sum_{i\in[k]} \lambda_i = 1$. We may assume without loss of generality that $\lambda_1 \geq \ldots \geq \lambda_k$. Define a solution $x(s)$ by

$$x(s) := \sum_{i\in[s-1]} \lambda_i x^i + \left(\sum_{i=s}^{k} \lambda_i\right) x^s, \tag{3}$$

then $x(s) \in X(s)$ and therefore $\mathrm{opt}(s) \leq \max_{c\in U} c^\top x(s)$. Furthermore let $c^*(s) \in \arg\max_{c\in U} c^\top x(s)$. It follows

$$\begin{aligned}
\mathrm{opt}(s) - \mathrm{opt}(k) &\leq \max_{c\in U} c^\top x(s) - \max_{c\in U} c^\top x^*(k) \\
&\leq c^*(s)^\top \left(x(s) - x^*(k)\right) \\
&= \sum_{i=s+1}^{k} \lambda_i c^*(s)^\top \left(x^s - x^i\right) \\
&\leq M(n) \left(\sum_{i=s+1}^{k} \lambda_i\right)
\end{aligned}$$

where the second inequality holds since $c^*(s)$ is a subgradient of the function $g(x) = \max_{c\in U} c^\top x$ in $x(s)$. For the first equality we used the definition of $x(s)$ and $x^*(k)$ and for the last inequality we used the assumption $\underline{p}(n) \leq \|x\|_0 \leq \bar{p}(n)$ and $m_\infty \leq c^*(s)_j \leq M_\infty$ for all $j \in [n]$. Due to the sorting $\lambda_1 \geq \ldots \geq \lambda_k$ and since the sum over all λ_i is one, we have $\lambda_i \leq \frac{1}{i}$ and hence

$$\begin{aligned}
M(n) \left(\sum_{i=s+1}^{k} \lambda_i\right) &\leq M(n) \left(\sum_{i=s+1}^{k} \frac{1}{i}\right) \\
&\leq M(n) \frac{k-s}{s+1}
\end{aligned}$$

which proves the result. □

Note that the bound derived in the latter lemma is small if s is large compared to the difference $k - s$. Particularly it holds

$$\mathrm{opt}(k) - \mathrm{opt}(k+1) \leq M(n) \frac{1}{k+1} \to 0 \quad \text{for } k \to \infty.$$

We can conclude that the objective value gain when k is increased by one unit decreases to zero when k gets large. From Lemma 3 we can easily derive an additive approximation guarantee for Algorithm 1.

Corollary 4. *For given k Algorithm 1 returns a solution to Problem* ($\mathrm{M}^3(k)$) *with*

$$\mathrm{approx}(k) \leq \mathrm{opt}(k') + M(n) \frac{n-k}{k+1}.$$

for all $k' \geq k$.

Proof. We can apply Lemma 3 for $k = n$ and $s = k$. The solution $x(s)$ constructed in (11) is composed of the same solutions $x^1, \ldots, x^s$ which Algorithm 1 returns. Hence the approximation guarantee holds also for this solution and we obtain

$$\mathrm{opt}(k) - \mathrm{opt}(n) \leq \mathrm{approx}(k) - \mathrm{opt}(n) \leq M(n) \frac{n-k}{k+1}.$$

It always holds $\mathrm{approx}(k) - \mathrm{opt}(k') \leq \mathrm{approx}(k) - \mathrm{opt}(n)$ for all $k' \geq k$ which proves the result. □

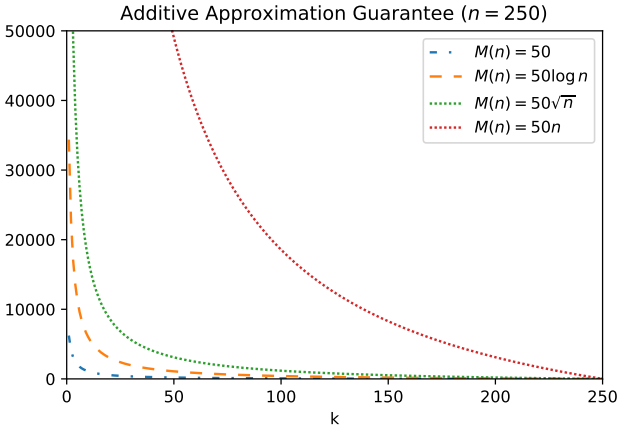

Figure 1: Additive approximation guarantee depending on k for fixed $n = 250$.

Note that the latter approximation guarantee holds for a large number of problem classes and the parameter $M(n)$ is problem-specific since it involves the functions $\underline{p}, \bar{p}$. While the approximation guarantee is linear in n and hence can be large, the interesting observation here is that it goes to zero if k approaches n.

In Figure 1 we show examples of the additive guarantee provided in the latter corollary for different constants $M(n)$. It can be seen that for all relevant functions $\underline{p}, \bar{p}$ with $1 \leq \underline{p}(n), \bar{p}(n) \leq n$ the approximation guarantee goes to zero when k approaches n. The smaller $\underline{p}, \bar{p}$ grow the faster the approximation guarantee approaches zero. This indicates that we can approximate $(\mathrm{M}^3(k))$ well by Algorithm 1 even for intermediate values k.

Ranges for k with given approximation guarantee The latter results lead to the intuition, that the range for k for which Algorithm 1 provides an approximation guarantee of $a(n)$, gets larger with increasing n. This is shown in the following lemma.

Lemma 5. *Let $k = n - l$ and $l \in [n-1]$, then Algorithm 1 returns a solution with approximation guarantee at most $a(n)$ for all*

$$l \in \left[0, \min\left\{n-1, \frac{a(n)n}{M(n)+a(n)}\right\}\right] \cap \mathbb{N}.$$

Proof. We can estimate

$$\mathrm{approx}(k) - \mathrm{opt}(k) \leq M(n)\frac{l}{n-l} \leq M(n)\frac{a(n)n}{M(n)+a(n)}\frac{M(n)+a(n)}{nM(n)} = a(n)$$

where the first inequality follows from Corollary 4 and the second follows from

$$l \leq \frac{a(n)n}{M(n)+a(n)}.$$

□

Note that the upper bound term $\frac{a(n)n}{M(n)+a(n)}$ for l in Lemma 5 (and therefore the absolute number of approximable $k < n$) goes to infinity if the approximation guarantee grows at least

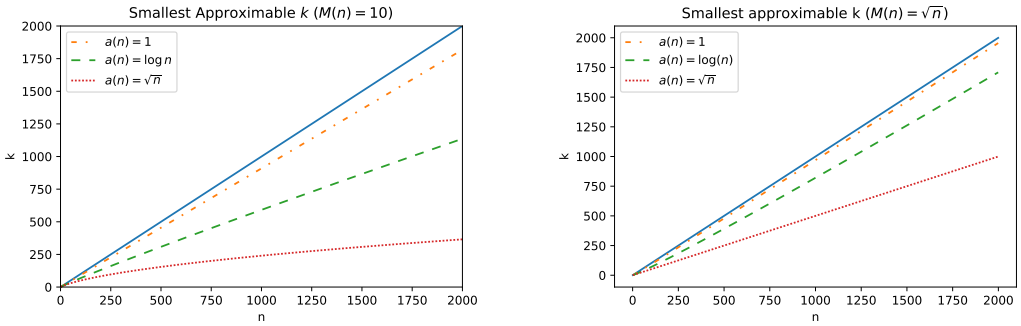

Figure 2: The smallest approximable $k = n-l$ for given approximation factor $a(n)$ and $M(n) = 10$ (left) or $M(n) = \sqrt{n}$ (right). The blue permanent line is the function $f(n) = n$, hence all k values between this line and the $k = n - l$ line are approximable within the given approximation guarantee.

as fast as $M(n)$, i.e. either $\frac{a(n)}{M(n)} \to \infty$ or $\frac{a(n)}{M(n)} \to C \in \mathbb{R}$ for $n \to \infty$. Additionally if $a(n)$ is constant, the upper bound goes to infinity if $M(n)$ grows sub-linearly, i.e. $\frac{n}{M(n)} \to \infty$ for $n \to \infty$. Note that the latter is the case for all examples in Proposition 2. This means that the range for k for which $\text{opt}(k)$ can be approximated grows with n. Additionally this means that the difficult instances, which do not achieve the approximation guarantee are the ones for $k < n - l$. In Figure 3.1 we show examples of the development of the ranges of k for which the approximation guarantee is ensured. It can be seen that the larger the approximation guarantee the smaller is the smallest k that can be approximated. Furthermore for increasing n the range of approximable k increases.

In Table 1 we show analytical lower bounds k_0 such that for all $k \geq k_0$ the given approximation guarantee is achieved. The values are derived as follows: first we estimate

$$M(n) = M_\infty \bar{p}(n) - m_\infty \underline{p}(n) \leq M_\infty \bar{p}(n).$$

Using this estimation we obtain that the result of Lemma 5 is valid for all

$$k \geq n - \frac{a(n)n}{M_\infty \bar{p}(n) + a(n)}.$$

Note that while the estimation of $M(n)$ results in easier notation, the size of this value significantly influences the quality of the bounds. Hence using the exact value for $M(n)$ can lead to better bounds. The values in Table 1 are derived by plugging in $\bar{p}(n)$ and $a(n)$ and bounding the derived terms from above. It is important to note that for each value k_0 in the table, it is given as a fraction of n, i.e. $k_0 = q(n)n$ where $q(n) \in (0, 1)$. Depending on the combination of $a(n)$ and $\bar{p}(n)$ either $q(n)$ is constant, $q(n) \to 0$ or $q(n) \to 1$ for $n \to \infty$. Note that a constant $q(n)$ means that the absolute number of k values for which the approximation holds grows with n but remains always a certain fraction of n. The case $q(n) \to 0$ is the most desirable case, since here the fraction of approximable k even increases. The less desirable case is the one where $q(n) \to 1$ since it means that the fraction of n which can be approximated goes to zero with increasing n. However for most bounds it holds that the absolute number of approximable k, i.e. $|[n - k_0, n] \cap \mathbb{N}|$, goes to infinity for $n \to \infty$.

Note that most of the problems considered in Proposition 2 have either $\bar{p}(n) = v$, $\bar{p}(n) = \sqrt{n}$ or $\bar{p}(n) = n$.

$\bar{p}(n)$ \ $a(n)$	ε	$\log n$	$n^{1-\gamma}$
const. v	$n\left(1-\frac{\varepsilon}{M_\infty v+\varepsilon}\right)$	$n\left(1-\frac{\log(n)}{M_\infty v+\log(n)}\right)$	$n\left(1-\frac{n^{1-\gamma}}{M_\infty v+n^{1-\gamma}}\right)$
$\log n$	$n\left(1-\frac{\varepsilon}{(M_\infty+1)\log n}\right)$	$n\left(1-\frac{1}{M_\infty+1}\right)$	$n\left(1-\frac{n^{\frac{1}{2}-\gamma}}{M_\infty+n^{\frac{1}{2}-\gamma}}\right)$
$n^{1-\delta}$	$n\left(1-\frac{\varepsilon}{n^{1-\delta}(M_\infty+1)}\right)$	$n\left(1-\frac{\log n}{n^{1-\delta}(M_\infty+1)}\right)$	$n\left(1-\frac{n^{\delta-\gamma}}{M_\infty+n^{\delta-\gamma}}\right)$

Table 1: For each pair of function $\bar{p}(n)$ and approximation guarantee $a(n)$ the table shows a value k_0 such that for all $k \geq k_0$ Algorithm 1 returns a solution of $(\mathrm{M}^3(k))$ with approximation guarantee $a(n)$. It holds $\gamma, \delta \in [0, 1)$.

Complexity We will now analyze the complexity of $(\mathrm{M}^3(k))$ for $k = n - l$ where $l \in [n-1]$ is a fixed parameter. The following lemma shows, that for a fixed l we can find a value n_0 such that for all $n \geq n_0$ Algorithm 1 returns an optimal solution (up to an additive accuracy of $\varepsilon > 0$) if $\bar{p}(n)$ grows sub-linear.

Lemma 6. *Assume $\bar{p}(n) \leq Cn^{1-\delta}$ where $\delta \in (0, 1]$ and $C > 0$. Furthermore let $k = n - l$ and $\varepsilon > 0$. If*

$$n \geq l^{\frac{1}{\delta}} \left(\frac{CM_\infty}{\varepsilon} + 1\right)^{\frac{1}{\delta}} \tag{4}$$

then $\mathrm{opt}(n) \leq \mathrm{opt}(n-l) \leq \mathrm{opt}(n) + \varepsilon$.

Proof. The inequality $\mathrm{opt}(n) \leq \mathrm{opt}(n-l)$ follows since $(\mathrm{M}^3(k))$ attains smaller optimal values if more solutions are allowed. To prove the inequality $\mathrm{opt}(n-l) \leq \mathrm{opt}(n) + \varepsilon$ we apply Lemma 3 with $\bar{p}(n) = Cn^{1-\delta}$, $\underline{p}(n) \equiv 0$, $s = n - l$ and $k = n$ and we obtain

$$\begin{aligned}
\mathrm{opt}(n-l) - \mathrm{opt}(n) &\leq Cn^{1-\delta} M_\infty \frac{l}{n-l+1} \\
&= CM_\infty \frac{l}{n^\delta - \frac{l}{n^{1-\delta}} + \frac{1}{n^{1-\delta}}} \\
&\leq CM_\infty \frac{l}{n^\delta - l} \\
&\leq \varepsilon
\end{aligned}$$

where in the second inequality we used $n \geq 1$ and in the third inequality we used inequality (4). □

The result of the latter lemma essentially says that, if we want to solve $(\mathrm{M}^3(k))$ with $k = n-l$, we do not have to worry about instances where n is large, since we can solve these instances efficiently by using Algorithm 1. This is a pretty surprising result since we only have to care about the instances with bounded n. If we fix all parameters of the bound in (4), then we can also solve $(\mathrm{M}^3(k))$ for small n by enumerating all solutions, which leads to the following theorem.

Theorem 7. *Assume $\bar{p}(n) \leq Cn^{1-\delta}$ where $\delta \in (0, 1]$ and $C > 0$. Furthermore let $k = n - l$ and $\varepsilon > 0$. Then we can calculate an optimal solution of $(\mathrm{M}^3(k))$ (up to an additive error of ε) in polynomial time in n, if all parameters M_∞, δ, C, ε and l are fixed.*

Proof. Given an instance of $(\mathrm{M}^3(k))$ we check if condition (4) is true or not, which can be done in polynomial time, since the right hand side is a constant. Note that to avoid calculations of the root-terms an easier bound without the exponents $\frac{1}{\delta}$ could be used.

Case 1: If the condition is true, we solve $M^3(n)$ up to an additive error of ε in polynomial time which can be done by Algorithm 1 due to Lemma 6 and Corollary 4.

Case 2: If condition (4) is not true, then n is bounded from above by the constant

$$\tau := l^{\frac{1}{\delta}} \left(\frac{2CM_\infty}{\varepsilon} + 1 \right)^{\frac{1}{\delta}}.$$

Therefore the number of possible solutions in X is in $\mathcal{O}(2^\tau)$ and hence the number of solutions of $(\mathrm{M}^3(k))$ is in $\mathcal{O}(2^{\tau k}) = \mathcal{O}(2^{\tau^2})$ since $k \leq n \leq \tau$. We can calculate an optimal solution in this case by enumerating all possible solutions of $(\mathrm{M}^3(k))$ and comparing the objective values. Note that we can calculate the objective value of a given solution $x^{(1)}, \ldots, x^{(k)}$ by solving the problem

$$\min_{x \in \operatorname{conv}\left(x^{(1)}, \ldots, x^{(k)}\right)} \max_{c \in U} c^\top x$$

which can again be done in polynomial time in n. □

The following corollary follows directly from Theorem 7 and Proposition 2.

Corollary 8. *Under the assumptions of Theorem 7 we can calculate an optimal solution of* $(\mathrm{M}^3(k))$ *(up to an additive error of* ε*) in oracle-polynomial time, if the underlying problem* (P) *is the spanning-tree problem on complete graphs, the matching problem on complete graphs, any cardinality constrained combinatorial problem, the TSP or the VRP.*

Note that the necessary condition on $\bar{p}$ is likely to be true for many other problems. Besides the p-selection problem several combinatorial problems with cardinality constraint $|x| = p$ were studied in the literature; see [13] for an overview. Finally note that the latter corollary states that we obtain an oracle-polynomial algorithm (and not a polynomial algorithm) which is because some of the stated problems are NP-hard and hence no polynomial algorithm can be derived unless $P = NP$.

3.2 Multiplicative approximation guarantees

In this subsection we follow similar ideas as in the latter section to derive multiplicative approximation guarantees. In contrast to the previous subsection we assume now that $k = \lceil qn \rceil$ for a fixed $q \in (0, 1)$. However all results also go through for the case $k = n - l$, leading to slightly different bounds and guarantees. With slight abuse of notation in the following we write $k = qn$ and assume that k is integer.

Approximation Guarantees The following lemma is the multiplicative counterpart of Lemma 3.

Lemma 9. *Let* $s, k \in [n]$ *where* $s < k$*, then it holds*

$$\operatorname{opt}(s) \leq \left(1 + \tilde{M}(n) \frac{k-s}{s+1}\right) \operatorname{opt}(k)$$

where $\tilde{M}(n) := \frac{M_\infty}{m_\infty} \frac{\bar{p}(n)}{\underline{p}(n)}$*. Furthermore if for all instances of dimension* n *it holds* $\underline{p}(n) = \bar{p}(n)$*, then* $\tilde{M}(n) := \frac{M_\infty}{m_\infty}$ *is independent of* n*.*

Proof. First note that due to the assumptions $\underline{p} \leq \|x\|_0$ for all $x \in X$ and $c \geq m_\infty \mathbf{1} > 0$ for all $c \in U$ we have $\text{opt}(k) > 0$ for all $k \in [n]$ and therefore the multiplicative approximation guarantee stated in the lemma is well defined.

By Lemma 3 we have

$$\text{opt}(s) \leq \text{opt}(k) + M(n)\frac{k-s}{s+1} \leq \text{opt}(k) + M_\infty \bar{p}(n)\frac{k-s}{s+1} \tag{5}$$

Let $x^*(k)$ be an optimal solution of Problem (1) with parameter k. Then by the results in [15] we have

$$\text{opt}(k) = \max_{c \in U} c^\top x^*(k)$$

and there exists a convex combination $x^*(k) = \sum_{i \in [k]} \lambda_i x^i$ where $x^i \in X$ and $\lambda \in \mathbb{R}^k_+$ with $\sum_{i \in [k]} \lambda_i = 1$. Then it holds

$$\text{opt}(k) = \max_{c \in U} c^\top x^*(k) \geq m_\infty \mathbf{1}^\top x^*(k) \tag{6}$$

where the last inequality holds due to the assumption $c \geq m_\infty \mathbf{1}$ and since $x^*(k) \geq 0$. We can now reformulate

$$m_\infty \mathbf{1}^\top x^*(k) = \sum_{i \in [k]} \lambda_i m_\infty \mathbf{1}^\top x^i \geq \sum_{i \in [k]} \lambda_i m_\infty \underline{p}(n) = m_\infty \underline{p}(n) \tag{7}$$

where in the first inequality we used the assumption $\underline{p}(n) \leq \|x\|_0$. Together with (6) we obtain $\text{opt}(k) \geq m_\infty \underline{p}(n)$. It follows

$$\frac{\text{opt}(s)}{\text{opt}(k)} \leq \frac{\text{opt}(k) + M_\infty \bar{p}(n)\frac{k-s}{s+1}}{\text{opt}(k)} \leq 1 + \frac{M_\infty \bar{p}(n)\frac{k-s}{s+1}}{m_\infty \underline{p}(n)}$$

which proves the result.

The second result follows directly from $\underline{p}(n) = \bar{p}(n)$. □

Analogously to Corollary 4 we can now derive a multiplicative approximation guarantee for each $k \in [n]$ from Lemma 9.

Corollary 10. *For given $k \in [n]$ Algorithm 1 returns a solution to Problem* ($\text{M}^3(k)$) *with multiplicative approximation guarantee*

$$\text{approx}(k) \leq \left(1 + \tilde{M}(n)\frac{n-k}{k+1}\right) \text{opt}(k')$$

for all $k' \geq k$.

Note that the behavior of the multiplicative approximation guarantee $a(n) = \tilde{M}(n)\frac{n-k}{k+1}$ is up to the factor $\tilde{M}(n)$ the same as for the additive approximation guarantee; see Figure 1 for exemplary behaviors.

If we choose k to be a fraction of n, i.e. $k = \lceil qn \rceil$ for a fixed $q \in (0,1)$, and if additionally we have $\underline{p}(n) = \bar{p}(n)$ (which is the case for the spanning tree problem and the matching problem on complete graphs, the traveling salesmen problem and each cardinality constrained problem), then the approximation factor is given as

$$1 + \frac{M_\infty}{m_\infty}\frac{n - \lceil qn \rceil}{\lceil qn \rceil + 1} \leq 1 + \frac{M_\infty}{m_\infty}\frac{(1-q)n}{qn} = 1 + \frac{M_\infty}{m_\infty}\frac{1-q}{q}$$

which is independent of the dimension n. This proves the following corollary.

Corollary 11. *Algorithm 1 has a constant multiplicative approximation guarantee for* $(\mathrm{M}^3(k))$ *if* $k = \lceil qn \rceil$ *for a fixed* $q \in (0,1)$ *and* $\underline{p}(n) = \bar{p}(n)$.

Note that for budgeted uncertainty sets where each parameter is allowed to deviate from its mean by at most 50% we have $\frac{M_\infty}{m_\infty} = 1.5$. If we choose $q = \frac{1}{2}$ the constant approximation factor of Algorithm 1 is

$$1 + \frac{M_\infty}{m_\infty}\frac{1-q}{q} = 2.5.$$

Ranges for k with given approximation guarantee The latter results lead to the intuition, that the range for k for which Algorithm 1 provides a multiplicative approximation guarantee of $a(n)$, gets larger with increasing n. This is shown in the following lemma.

Lemma 12. *Let* $k = qn$ *and* $q \in (0,1]$*, then Algorithm 1 returns a solution with multiplicative approximation guarantee at most* $a(n)$ *for all*

$$q \in \left[\frac{\tilde{M}(n)}{\tilde{M}(n) + a(n)}, 1\right].$$

Proof. We can estimate

$$\begin{aligned}\frac{\text{approx}(k)}{\text{opt}(k)} &\leq 1 + \tilde{M}(n)\frac{n-qn}{qn} \leq 1 + \tilde{M}(n)\frac{1-q}{q} \\ &\leq 1 + \tilde{M}(n)\frac{a(n)}{\tilde{M}(n)+a(n)}\frac{\tilde{M}(n)+a(n)}{\tilde{M}(n)} = 1 + a(n)\end{aligned}$$

where the first inequality follows from Corollary 10 and the third follows from

$$q \geq \frac{\tilde{M}(n)}{\tilde{M}(n) + a(n)}. \tag{8}$$

□

Note that the lower bound term (8) for q goes to zero if the approximation guarantee grows faster than $\tilde{M}(n)$ while it goes to one if it grows slower. If $\tilde{M}(n)$ grows as fast as $a(n)$, then it is constant. In Figure 3 we show examples for the development of the ranges for k for which the approximation guarantee is ensured.

In Table 2 we show lower bounds k_0 such that for all $k \geq k_0$ the given approximation guarantee is achieved. The values are derived analogously to Table 1. It is important to note that each value k_0 in the table is given as a fraction of n, i.e. $k_0 = q(n)n$ where $q(n) \in (0,1)$. Depending on the combination of $a(n)$ and $\frac{\bar{p}(n)}{\underline{p}(n)}$ either $q(n)$ is constant, $q(n) \to 0$ or $q(n) \to 1$ for $n \to \infty$. Note that a constant $q(n)$ means that the number of k values for which the approximation holds grows with n but remains always a certain fraction of n. The case $q(n) \to 0$ is the most desirable case, since here the number of k values for which the approximation holds grows with n and the fraction of values for which it holds even increases. The less desirable case is the one where $q(n) \to 1$ since it means that the fraction of n which can be approximated goes to zero with increasing n. However for all bounds the absolute number of approximable k values, i.e. $|[n - k_0, n] \cap \mathbb{N}|$ goes to infinity for $n \to \infty$. Note that in contrast to the results in Section 3.1 here the function ratio $\frac{\bar{p}(n)}{\underline{p}(n)}$ has to be bounded. As shown in Proposition 2 this fraction is constant for the spanning tree problem and the matching problem on complete graphs, as well as for all other mentioned problems.

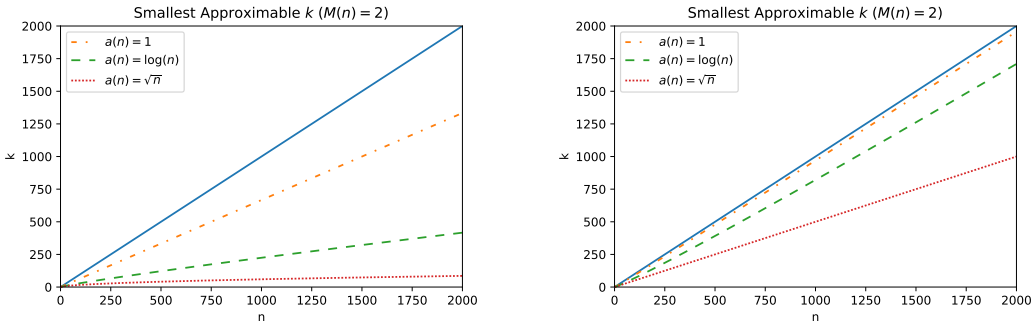

Figure 3: The smallest approximable $k = qn$ for given approximation factor $a(n)$ and $\tilde{M}(n) = 2$ (left) or $\tilde{M}(n) = \sqrt{n}$ (right). The blue permanent line is the function $f(n) = n$, hence all k values between this line and the $k = qn$ line are approximable within the given approximation guarantee.

$\frac{\bar{p}(n)}{\underline{p}(n)}$ \ $a(n)$	ε	$\log n$	$n^{1-\gamma}$
const. v	$n\left(\frac{\hat{m}v}{\hat{m}v+\varepsilon}\right)$	$n\left(\frac{\hat{m}v}{\hat{m}v+\log n}\right)$	$n\left(\frac{\hat{m}v}{\hat{m}v+n^{1-\gamma}}\right)$
$\log n$	$n\left(\frac{\hat{m}\log n}{\hat{m}\log n+\varepsilon}\right)$	$n\left(\frac{\hat{m}}{\hat{m}+1}\right)$	$n\left(\frac{\hat{m}\log n}{\hat{m}\log n+n^{1-\gamma}}\right)$
$n^{1-\delta}$	$n\left(\frac{\hat{m}n^{1-\delta}}{\hat{m}n^{1-\delta}+\varepsilon}\right)$	$n\left(\frac{\hat{m}n^{1-\delta}}{\hat{m}n^{1-\delta}+\log n}\right)$	$n\left(\frac{\hat{m}}{\hat{m}+n^{\delta-\gamma}}\right)$

Table 2: For each pair of function ratio $\frac{\bar{p}(n)}{\underline{p}(n)}$ and approximation guarantee $a(n)$ the table shows a value k_0 such that for all $k \geq k_0$ Algorithm 1 returns a solution of $(\mathrm{M}^3(k))$ with approximation guarantee $a(n)$. It holds $\hat{m} := \frac{M_\infty}{m_\infty}$ and $\gamma, \delta \in [0, 1)$.

Complexity Unfortunately following the previous methods it is not possible to prove a multiplicative version of Theorem 7 which provides a polynomial time algorithm to approximate $(\mathrm{M}^3(k))$ for $k = qn$ for each fixed $q \in (0, 1)$. However a weaker version of the result will be proved in the following.

Lemma 13. *Assume that* $\underline{p}(n) = \bar{p}(n)$ *and let* $\varepsilon > 0$, $k = qn$. *If*

$$q \in \left[\frac{\frac{M_\infty}{m_\infty}}{\varepsilon + \frac{M_\infty}{m_\infty}}, 1\right]$$

then $\mathrm{opt}(qn) \leq (1+\varepsilon)\,\mathrm{opt}(n)$.

Proof. We can apply Lemma 9 with $k = n$ and $s = qn$ which leads to the bound

$$\mathrm{opt}(qn) \leq \left(1 + \tilde{M}(n)\frac{1-q}{q}\right)\mathrm{opt}(n).$$

where $\tilde{M}(n) = \frac{M_\infty}{m_\infty}$ since $\underline{p}(n) = \bar{p}(n)$. We can estimate

$$\tilde{M}(n)\frac{1-q}{q} \leq \frac{M_\infty}{m_\infty}\frac{\varepsilon}{\varepsilon + \frac{M_\infty}{m_\infty}}\frac{\varepsilon + \frac{M_\infty}{m_\infty}}{\frac{M_\infty}{m_\infty}} = \varepsilon$$

□

Note that the latter result only holds if $\underline{p}(n) = \bar{p}(n)$. However this is the case for all problems in Proposition 2 on complete graphs. Furthermore we do not need any assumption on the behavior of $\bar{p}(n)$ here. The following theorem follow directly from Lemma 13.

Theorem 14. *Assume that for all $n \in \mathbb{N}$ it holds $\underline{p}(n) = \bar{p}(n)$, let $\varepsilon > 0$ and $k = qn$ with*

$$q \in \left[\frac{\frac{M_\infty}{m_\infty}}{\varepsilon + \frac{M_\infty}{m_\infty}}, 1 \right].$$

Then $(\mathrm{M}^3(k))$ can be approximated with an multiplicative error of $1+\varepsilon$ in oracle-polynomial time in n if all parameters $\frac{M_\infty}{m_\infty}$ and ε are fixed.

The next corollary follows directly from Theorem 14 and Proposition 2.

Corollary 15. *Under the assumptions of Theorem 14 Algorithm 1 calculates a solution of $(\mathrm{M}^3(k))$ with multiplicative approximation guarantee $1 + \varepsilon$ in oracle-polynomial time for the spanning-tree problem on complete graphs, the matching problem on complete graphs, any cardinality constrained combinatorial problem, the TSP or the VRP.*

4 Approximation algorithm for k-adaptability problems

In this section we extend the results of the previous section to the k-adaptability problem and answer the following research questions:

RQ1. Can we derive general additive and multiplicative approximation guarantees for the k-adaptability problem depending on k?

RQ2. For which range of k does the k-adaptability problem provide a certain approximation guarantee for the two-stage robust problem?

RQ3. Can we derive an algorithm which efficiently calculates a solution of the k-adaptability problem with a certain approximation guarantee?

We already showed in Section 2 that the optimal values of the k-adaptability problem follow the chain

$$\mathrm{adapt}(1) \geq \mathrm{adapt}(2) \geq \ldots \geq \mathrm{adapt}(n) \geq \mathrm{opt}(2RO) = \mathrm{adapt}(n+1) = \mathrm{adapt}(n+2) = \ldots$$

where adapt(1) is equal to the optimal value of the classical robust problem.

However calculating a set of $n+1$ second-stage policies can be computationally heavy. Hence it would be desirable to find smaller bounds for k, which lead to a certain approximation guarantee for the optimal value of the two-stage robust problem $\mathrm{opt}(2RO)$. In the following we will show that the results from the previous section can be extended to (k-adapt) to provide better bounds on k. We will first show that Lemma 3 and Lemma 9 also hold for (k-adapt).

Similar to the previous sections, we assume that for a given problem class there exist functions $\underline{p}, \bar{p} : \mathbb{N} \to \mathbb{R}_+$ such that for each instance $(X, Y(X))$ of the problem class where the number of second-stage variables is n it holds $\underline{p}(n) \leq \|y\|_0 \leq \bar{p}(n)$ for all $y \in Y(x)$ for all $x \in X$. We also assume again that $\|\xi\|_\infty \leq M_\infty$ and $\|\xi\|_\infty \geq m_\infty > 0$ for all $\xi \in U$. Furthermore we assume $d \geq 0$.

Lemma 16. *Assume $s, k \in [n+1]$ where $s < k$, then it holds*

$$\text{adapt}(s) - \text{adapt}(k) \leq M(n)\frac{k-s}{s+1}.$$

where $M(n) := M_\infty \bar{p}(n) - m_\infty \underline{p}(n)$.

The proof of Lemma 16 is very similar to the proof of Lemma 3 and is provided in the Appendix.

We now prove that also Lemma 9 can be extended to the k-adaptable case.

Lemma 17. *Let $s, k \in [n]$ where $s < k$ and for the first-stage costs it holds $d \geq 0$, then it holds*

$$\text{adapt}(s) \leq \left(1 + \tilde{M}(n)\frac{k-s}{s+1}\right)\text{adapt}(k)$$

where $\tilde{M}(n) := \frac{M_\infty}{m_\infty}\frac{\bar{p}(n)}{\underline{p}(n)}$. Furthermore if for all instances of dimension n it holds $\underline{p}(n) = \bar{p}(n)$, then $\tilde{M}(n) := \frac{M_\infty}{m_\infty}$ is independent of n.

Proof. Similar as in the proof of Lemma 9 we can estimate

$$\text{adapt}(k) \geq m_\infty \underline{p}(n)$$

where we used the fact that the first stage costs $d^\top x \geq 0$. We can now follow the proof of Lemma 9 to prove the result. □

Approximation guarantees for two-stage robust optimization We can now follow the derivations in Section 3.1 and 3.2 analogously to prove the following bounds on the number of second-stage policies needed to approximate the optimal value of the robust two-stage robust problem up to a certain approximation guarantee.

Lemma 18. *Let $k = n - l$, then*

$$\text{adapt}(k) \leq \text{opt}(2RO) + a(n)$$

for all

$$l \in \left[0, \min\left\{n-1, \frac{a(n)n}{M(n)+a(n)}\right\}\right] \cap \mathbb{N}.$$

From the latter Lemma we can conclude that to derive a solution to (2RO) with an additive approximation guarantee of $a(n)$ it is enough to calculate an optimal solution of (k-adapt) with

$$k = n - \frac{a(n)n}{M(n)+a(n)}$$

second-stage policies. Furthermore all the bounds k_0 provided in Table 1 can also be used in this case for (k-adapt). Note that the number of second-stage policies needed in Table 1 is significantly smaller than n, especially with increasing n. This is also shown in Example 20 and 21.

Analogously as in the latter derivation we can conclude for the multiplicative approximation guarantees.

Lemma 19. *Let $k = qn$ and $q \in (0, 1]$, then*

$$\text{adapt}(k) \leq (1 + a(n)) \,\text{opt}(2RO)$$

for all

$$q \in \left[\frac{\tilde{M}(n)}{\tilde{M}(n) + a(n)}, 1 \right].$$

From the latter Lemma we can conclude that to derive a solution to (2RO) with an multiplicative approximation guarantee of $a(n)$ it is enough to calculate an optimal solution of (k-adapt) with

$$k = \lceil \frac{\tilde{M}(n)}{\tilde{M}(n) + a(n)} n \rceil$$

second-stage policies. Furthermore all the bounds k_0 provided in Table 2 can also be used in this case for (k-adapt). Note that the number of second-stage policies needed in Table 2 is significantly smaller than n, especially with increasing n. This is shown in the following examples.

Example 20 (Facility Location)**.** *Consider the two-stage robust facility location problem (see [32]) for a fixed number s of locations and a variable number r of customers. Fixing the number of locations is rarely a restriction in practice since the number of locations can be chosen to be small. We consider the case $s = 100$. In the first stage we can open facilities in a subset of the locations which comes with a cost. In the second-stage we have to assign each customer to exactly one of the opened facilities. We assume that the costs of assigning customers to facilities is uncertain. In the classical integer programming formulation the first-stage dimension is $m = s$ and the second-stage dimension is $n = sr$. For each second-stage solution y it holds $\|y\|_0 = r$, i.e. $\underline{p}(n) = \bar{p}(n) = \frac{1}{s}n$.*

We assume the second-stage assignment costs are given by an uncertainty set $U \subseteq [\bar{c} - 0.02\bar{c}, \bar{c} + 0.02\bar{c}]^n$ where $\bar{c} \in \mathbb{R}^n$ is a given mean vector with entries in $[800, 1000]$. Then we have

$$784 \leq \|c\|_\infty \leq 1020 \quad \forall\ c \in U$$

and hence

$$\frac{M_\infty}{m_\infty} \leq \frac{1020}{784} \leq 2.$$

We can conclude that $M(n) \leq 2.5n$ and $\tilde{M}(n) \leq 2$. Applying the result of Lemma 12 we obtain the following number of second-stage policies which have to be used in the k-adaptability approach to achieve a additive or multiplicative approximation guarantee of $a(n)$ for (2RO). In brackets we show the fraction of n for $r = 1000$ customers.

$a(n)$	additive guarantee	multiplicative guarantee
1	$0.999996n$	$\frac{2}{3}n$
$\log(n)$	$0.99998n$	$0.15n$
$\sqrt{n}$	$0.9987n$	$0.007n$

Table 3: Number of second-stage policies needed in the k-adaptability approach (with $n \geq 1000$) to approximate the recoverable robust optimization problem up to an approximation guarantee of $a(n)$.

Example 21 (Recoverable Robust Optimization)**.** *In the recoverable robust optimization problem (see [36]) the idea is to find a solution in $X \subseteq \{0,1\}^m$ which is worst-case optimal where in the second-stage after the uncertain parameters are revealed we are allowed change at most $\bar{p}$ entries of the solution and the result has to be another feasible solution in X. This can be modeled as a two-stage robust optimization problem as*

$$\min_{x \in X} \max_{c \in U} \min_{y \in Y(x)} c^\top y$$

where

$$Y(x) : \left\{ z^+, z^- \in \{0,1\}^m : x + z^+ - z^- \in X,\ \sum_{i=1}^{m} z_i^+ + z_i^- \leq \bar{p},\ \right\}.$$

Here the dimension of the second-stage is $n = 2m$ and $\underline{p}(n) = \bar{p}(n) = \bar{p}$. We consider the same uncertainty set as in Example 20, i.e.

$$M(n) = M_\infty \bar{p} - m_\infty \bar{p} = 0.04 \|\bar{c}\|_\infty \bar{p} \leq 40\bar{p}$$

and $\tilde{M}(n) \leq 2$. In the following table we show the values for the number of second-stage policies k we have to use in the k-adaptability approach to get an additive (or multiplicative) approximation guarantee of $a(n)$. We show the values for $m = 500$ (i.e. $n = 1000$) and $\bar{p} = 10$. Note that the values also holds for dimensions larger than that, however in this case the fraction should be re-calculated to obtain better values.

The result indicates, e.g. if we use the k-adaptability approach with $k = 60$ second-stage policies, then we obtain a multiplicative $\sqrt{n}$-approximation of the recoverable robust problem.

$a(n)$	additive guarantee	multiplicative guarantee
1	$0.998n$	$\frac{2}{3}n$
$\log(n)$	$0.98n$	$0.23n$
$\sqrt{n}$	$0.93n$	$0.06n$

Table 4: Number of second-stage policies needed in the k-adaptability approach (with $n \geq 1000$) to approximate the recoverable robust optimization problem up to an approximation guarantee of $a(n)$.

Furthermore note that Corollary 11 can directly be extended to k-adaptability problems.

Corollary 22. *The k-adaptability problem achieves a constant multiplicative approximation guarantee of*

$$1 + \frac{M_\infty}{m_\infty} \frac{1-q}{q}$$

for the two-stage robust optimization problem if $k = \lceil qn \rceil$ for a fixed $q \in (0,1)$ and $\underline{p}(n) = \bar{p}(n)$.

Note that the latter corollary is especially valid for the recoverable robust problem. For budgeted uncertainty sets where each parameter is allowed to deviate from its mean by at most 50% we have $\frac{M_\infty}{m_\infty} = 1.5$. If we choose $q = \frac{1}{2}$ the constant approximation factor of Algorithm 1 is

$$1 + \frac{M_\infty}{m_\infty} \frac{1-q}{q} = 2.5.$$

4.1 Approximation algorithm for k-adaptability problems

The results in the latter subsection provide useful bounds on the number of second-stage policies k which have to be calculated to obtain a certain approximation guarantee for the exact two-stage robust problem. While this is useful from a theoretical point of view, there is no algorithm known which can achieve these approximation guarantees. Unfortunately using Algorithm 1 is not possible since it cannot provide first-stage solutions.

In [26] the authors derive approximation guarantees for the two-stage robust problem with discrete uncertainty. They show that by choosing an appropriate set of scenarios and solving the two-stage robust problem for this set, the obtained first-stage solution has a certain approximation guarantee. In the following we will adapt the results of [26] to derive an approximation algorithm for the k-adaptability problem with convex uncertainty sets.

We first adapt and generalize the result from [26] to our setting, i.e. for convex uncertainty sets and for the k-adaptability problem.

Lemma 23. *Let $\{\hat{\xi}^1, \dots, \hat{\xi}^t\} \subset U$ be a set of scenarios and $\alpha \geq 1$ such that*

$$\forall \xi \in U \ \exists i \in [t] : \ \xi \leq \alpha \hat{\xi}^i \tag{9}$$

holds. Then any optimal solution $\hat{x}$ of the problem

$$\begin{aligned} \min \ & d^\top x + \mu \\ s.t. \quad & \mu \geq (y^i)^\top \hat{\xi}^i \quad i = 1, \dots, t \\ & x \in X, \ y^1, \dots, y^t \in Y(x) \end{aligned} \tag{10}$$

has a multiplicative approximation guarantee of $\alpha - 1$ for the k-adaptability problem for all $k \leq t$.

Proof. The proof follows similar steps as in [26]. Let $\hat{x}, \hat{y}^1, \dots, \hat{y}^t$ be an optimal solution of Problem (10) and let $x^*, y^{1*}, \dots, y^{k*}$ be an optimal solution of the k-adaptability problem (k-adapt). We bound the k-adaptable objective value of solution $\hat{x}, \hat{y}^1, \dots, \hat{y}^t$ as follows:

$$\begin{aligned} d^\top \hat{x} + \max_{\xi \in U} \min_{i=1,\dots,t} \xi^\top \hat{y}^i &\leq d^\top \hat{x} + \max_{\xi \in \{\hat{\xi}^1, \dots, \hat{\xi}^t\}} \min_{i=1,\dots,t} \alpha \xi^\top \hat{y}^i \\ &\leq \alpha \left(d^\top \hat{x} + \max_{\xi \in \{\hat{\xi}^1, \dots, \hat{\xi}^t\}} \min_{i=1,\dots,t} \xi^\top \hat{y}^i \right) \\ &\leq \alpha \left(d^\top x^* + \max_{\xi \in \{\hat{\xi}^1, \dots, \hat{\xi}^t\}} \min_{i=1,\dots,k} \xi^\top y^{i*} \right) \\ &\leq \alpha \left(d^\top x^* + \max_{\xi \in U} \min_{i=1,\dots,k} \xi^\top y^{i*} \right) \\ &= \alpha \cdot \text{adapt}(k) \end{aligned}$$

where for the first inequality we used Condition (9) and for the second inequality we used $\alpha \geq 1$ and $d^\top \hat{x} \geq 0$. Note that Problem (10) is equivalent to the problem

$$\min_{\substack{x \in X \\ y^1, \dots, y^t \in Y(x)}} \ d^\top x + \max_{\xi \in \{\hat{\xi}^1, \dots, \hat{\xi}^t\}} \min_{i=1,\dots,t} \xi^\top y^i$$

and since $\hat{x}, \hat{y}^1, \dots, \hat{y}^t$ is an optimal solution of the latter problem the third inequality follows since $k \leq t$. The last inequality follows since $\{\hat{\xi}^1, \dots, \hat{\xi}^t\} \subset U$. □

The latter lemma shows how to find a solution with an approximation guarantee of $\alpha - 1$ for the k-adaptability problem. However, deriving the scenario set $\{\hat{\xi}^1, \dots, \hat{\xi}^t\}$ which leads to the best approximation guarantee is computationally heavy; see [26]. In the following we propose a more efficient heuristic approach in Algorithm 2 where an arbitrary set of t scenarios in U is used, which leads to a certain approximation guarantee for the k-adaptability problem; see Theorem 25.

Algorithm 2 Approximation Algorithm K-adaptability

Input: $n \in \mathbb{N}$, $k \in [n]$, $t \geq k$, convex set $U \subset \mathbb{R}^n$, $\underline{u}, \bar{u} \in \mathbb{R}^n$ s.t. $U \subseteq [\underline{u}, \bar{u}]$

1: Select arbitrary $\hat{\xi}^1, \dots, \hat{\xi}^t \in U$.
2: Calculate an optimal solution $(\hat{x}, \hat{y}^1, \dots, \hat{y}^t)$ of Problem (10) with scenarios $\hat{\xi}^1, \dots, \hat{\xi}^t$.
3: Calculate solutions $\bar{y}^1, \dots, \bar{y}^k \in Y(\hat{x})$ by Algorithm 1 for the min-max-min problem

$$d^\top \hat{x} + \min_{y^1,\dots,y^k \in Y(\hat{x})} \max_{\xi \in U} \min_{i=1,\dots,k} \xi^\top y^i$$

4: **Return:** $\hat{x}, \bar{y}^1, \dots, \bar{y}^k$

Remark 24 (Variant of Algorithm 2)**.** *In Remark 1 we presented a variant of Algorithm 1 where for the selection of the k solutions a sparsity constraint is added to the Problem in Step 2 of Algorithm 1. Obviously we directly obtain a variant of Algorithm 2 by applying Algorithm 1 (Variant) in Step 3 of Algorithm 2. We will denote this algorithm as Algorithm 2 (Variant).*

In the following we define the k-adaptable objective value for the solution $\hat{x}, \bar{y}^1, \dots, \bar{y}^k$ returned by Algorithm 2 as

$$\text{approx}_{\text{adapt}}(k) := d^\top \hat{x} + \max_{\xi \in U} \min_{i=1,\dots,k} \xi^\top \bar{y}^i.$$

We can now show the following theorem.

Theorem 25. *Let $k \in \mathbb{N}$ with $k \leq n$, $U \subset \mathbb{R}^n_+$ and $\underline{u}, \bar{u} \in \mathbb{R}^n_+$ such that $U \subseteq [\underline{u}, \bar{u}]$. Then for the solution $\hat{x}, \bar{y}^1, \dots, \bar{y}^k$ returned by Algorithm 2 it holds*

$$\text{approx}_{adapt}(k) \leq \alpha \left(1 + \tilde{M}(n) \frac{n-k}{k+1}\right) \text{adapt}(k)$$

where $\alpha = \max_{j=1,\dots,n} \frac{\bar{u}_j}{\underline{u}_j}$ and $\tilde{M}(n)$ is defined as in Lemma 9.

Proof. We first derive the approximation value α in Lemma 23 for the set of scenarios $\hat{\xi}^1, \dots, \hat{\xi}^t$ selected in Step (1). Note that for every scenario $\xi \in U$ it holds

$$\frac{\xi_j}{\hat{\xi}^i_j} \leq \frac{\bar{u}_j}{\underline{u}_j} \quad \forall i \in [t], j \in [n]$$

since $\hat{\xi}^1, \dots, \hat{\xi}^t \in U$. Hence, the assumption of Lemma 23 is true for

$$\alpha := \max_{j=1,\dots,n} \frac{\bar{u}_j}{\underline{u}_j}.$$

We can now bound the objective value of the solution calculated by Algorithm 2 as follows:

$$\begin{aligned}
\text{approx}_{\text{adapt}}(k) &= d^\top \hat{x} + \max_{\xi \in U} \min_{i=1,\dots,k} \xi^\top \bar{y}^i \\
&\leq d^\top \hat{x} + \left(1 + \tilde{M}(n)\frac{n-k}{k+1}\right) \min_{y^1,\dots,y^k \in Y(\hat{x})} \max_{\xi \in U} \min_{i=1,\dots,k} \xi^\top y^i \\
&\leq \left(1 + \tilde{M}(n)\frac{n-k}{k+1}\right) \left(d^\top \hat{x} + \min_{y^1,\dots,y^k \in Y(\hat{x})} \max_{\xi \in U} \min_{i=1,\dots,k} \xi^\top y^i\right) \\
&\leq \alpha \left(1 + \tilde{M}(n)\frac{n-k}{k+1}\right) \text{adapt}(k)
\end{aligned}$$

where the first inequality follows from the multiplicative approximation guarantee derived in Corollary 10 which we obtain since we apply Algorithm 1 in Step (3) to the min-max-min problem with fixed first-stage solution $\hat{x}$. The second inequality follows since $\left(1 + \tilde{M}(n)\frac{n-k}{k+1}\right) \geq 1$ and $d^\top \hat{x} \geq 0$ and the third inequality follows by applying Lemma 23 with the α value derived above. □

Note that the derived approximation guarantee differs only by the multiplicative factor α from the one shown in Lemma 17 and α does not depend on the dimension n. Hence the analysis shown in Section 3 can be easily adapted here. However, the approximation guarantee can never be smaller than α. The factor α is given as the maximum ratio of the upperbound and the lowerbound value of the uncertainty set over all dimensions. Hence, larger uncertainty sets lead to larger approximation guarantees. Note that if we consider budgeted uncertainty sets where the deviation of the mean-value is at most 50%, then $\alpha = \max_{j=1,\dots,n} \frac{\bar{u}_j}{\underline{u}_j} \approx 1.5$.

One limitation of our analysis is that the parameter α is independent of the number of scenarios t which are chosen. This is counter intuitive since generating more scenarios in Step 1 of Algorithm 2 can lead to better approximation guarantees.

4.2 Calculating lower bounds for k-adaptability problems

In the following we show how to efficiently calculate lower bounds for the k-adaptability problem which can be used together with the upper bounds calculated in the previous section in classical algorithmic frameworks to calculate optimality gaps.

Using the reformulation of the k-adaptability problem

$$\min_{\substack{x \in X,\ \lambda \geq 0 \\ y = \sum_{i \in [k]} \lambda_i y^i \\ \sum_{i \in [k]} \lambda_i = 1 \\ y^1,\dots,y^k \in Y(x)}} \max_{\xi \in U} d^\top x + \xi^\top y.$$

it is easy to see that a valid lower bound is given by the case where $k = n$ and in this case the problem is equivalent to

$$\min_{x \in X, y \in \text{conv}(Y(x))} \max_{\xi \in U} d^\top x + \xi^\top y.$$

This problem can be lower bounded by

$$\min_{(x,y) \in \text{conv}(Z)} \max_{\xi \in U} d^\top x + \xi^\top y$$

where $Z = \{(x, y) : x \in X, y \in Y(x)\}$. The latter problem is of the type (2) and hence it is equivalent to a min-max-min robust problem over the set Z with $k = n$. This can be efficiently solved again by the standard methods shown in [15]. We will use this lower bound to provide optimality gaps for Algorithm 2 in Section 5.

5 Computations

In this section we test Algorithm 1 (and its variant) on random instances from the min-max-min literature for the shortest path problem and the knapsack problem and compare the objective values to the lower bound obtained by solving $\mathrm{M}^3(n)$.

Additionally we test Algorithm 2 on a generic two-stage robust problem studied in [25] and on a two-stage version of the shortest path problem (see [27]), where in the first stage a set of edges has to be bought, while after the edge costs are revealed in the second-stage a shortest path has to be calculated where only edges bought in the first stage can be used. We apply Algorithm 2 with $t = n$ scenarios selected in the first step.

All algorithms were implemented in Python 3.10 and all optimization problems were solved by Gurobi 10 [31] with standard hyperparameter setup. Experiments were executed on a cluster with AMD Genoa 9654 (2x) 96 Cores/Socket 2.4GHz 360W CPU and with 24 x 16GiB 4800MHz, DDR5 RAM.

5.1 Instances

We test the algorithm on the **minimum knapsack problem (KP)** which is given by

$$\min_{\substack{a^\top x \geq b \\ x \in \{0,1\}^n}} \bar{c}^\top x$$

where the instances were generated as in [20, 28]. For $n \in \{50, 100\}$ we generate 10 random knapsack instances where the mean-costs $\bar{c}_i$ and the weights a_i were drawn from a uniform distribution on $\{1, \ldots, 100\}$ and the knapsack capacity b was set to 35% of the sum of all weights. For each knapsack instance we generated a budgeted uncertainty set

$$U = \left\{ c \in \mathbb{R}^n : c_i = \bar{c}_i + \delta_i \Delta_i, \sum_{i \in [n]} \delta_i \leq \Gamma, \delta \in [0,1]^n \right\}$$

where each Δ_i was drawn uniformly from $\{1, \ldots, \bar{c}_i\}$ with budget parameter $\Gamma \in \{2, 5, 10\}$.

For the **shortest path problem (SPP)** we use the original instances from [32] which were also used in several other publications of the min-max-min literature. We consider random graphs $G = (V, E)$ with $|V| \in \{30, 50\}$ nodes corresponding to points in the Euclidean plane with random coordinates in $[0, 10]$. For each dimension we randomly select 10 of the graphs generated in [32]. We consider budgeted uncertainty sets described as above where the mean values $\bar{c}_{ij}$ on edge (i, j) are set to the euclidean distance of node i and j and the deviation values are set to $\Delta_{ij} = \frac{\bar{c}_{ij}}{2}$. The budget parameter Γ was chosen in $\in \{2, 5, 10\}$.

For the two-stage case we consider the **generic k-adaptability problem (GP)** studied in [25] which is defined as

$$\begin{aligned} & \min \max_{\xi \in U} \min_{i=1,\ldots,k} d^\top x + \xi^\top y^i \\ s.t. \quad & \sum_{i=1}^{m} x_i = 10 \\ & f^\top y^i - a^\top x \geq 0 \quad i = 1, \ldots, k \\ & x \in \{0,1\}^m, y^1, \ldots, y^k \in \{0,1\}^n \end{aligned}$$

where x are the first-stage decision variables and y^i the second-stage decision variables and $n = m = 50$. The parameters d_i, a_i and f_i are uniformly and randomly generated in $[8, 12]$,

$[50, 100]$ and $[80, 90]$, respectively. We generate budgeted uncertainty sets as above with mean values $\bar{\xi}_i$ uniformly and randomly generated from $[8, 12]$ and deviations values $\Delta_i = \frac{\bar{\xi}_i}{4}$. The budget parameter Γ was chosen in $\in \{2, 5, 10\}$.

Finally we consider a **two-stage network construction variant of the shortest path problem (2SP)**. We consider the same instances as for the shortest path problem but study the k-adaptability problem

$$\begin{aligned} \min \max_{\xi \in U} \min_{i=1,\dots,k} & d^\top x + \xi^\top y^i \\ s.t. \quad & x \in \{0,1\}^E \\ & y^1, \dots, y^k \in Y_{\mathrm{SP}}(x) \end{aligned}$$

where $Y_{\mathrm{SP}}(x)$ is the set of paths in G (encoded by their 0-1 incidence vectors) where $y^i_e \leq x_e$ for all $i \in [k]$. Hence, the task is to buy a set of edges in the first-stage (indicated by the decisions x) such that for every scenario $\xi \in U$ a good path y^i exists which only uses edges bought in the first stage. This problem was already studied e.g. in [27]. The first-stage costs are set to $d = \frac{1}{2} \min_{i=1,\dots,n} \bar{\xi}_i \mathbf{1}$ where $\bar{\xi}$ is the mean-cost vector of the budgeted uncertainty set. The latter choice is motivated by the observation that if the costs are set significantly smaller, the optimal solution is to buy all edges of the graph. On the other hand if the value is set significantly larger the optimal solution is to buy just one path in the graph, so in the second-stage there is no option to adapt to the scenarios.

5.2 Computational Results

In the following subsections we show the achieved optimality gaps and runtimes of Algorithm 1 (and its variant) and Algorithm 2 (and its variant) for the problem instances introduced in the previous subsection and for different values of k. All values are given as the average over 10 random instances.

To obtain the optimality gap, we first run Algorithm 1 (or Algorithm 2) and record the objective value obj of the returned solution. Then we calculate a lowerbound for the KP and SPP instances by calculating the optimal value of the corresponding min-max-min problem for $k = n$ by applying the iterative method presented in Section 4.2 to the one-stage problems. For the two-stage problems GP and 2SP we apply the same algorithm presented in 4.2 to obtain a lower bound. The optimality gap is defined as

$$\text{gap} := \frac{\text{obj} - \text{lowerbound}}{\text{lowerbound}} \cdot 100.$$

The runtimes are given in seconds. We set a timelimit of 1800 seconds for the problem to be solved in Step 2 of Algorithm 1 (Variant) since in our experiments it was the only subproblem which sometimes lead to huge runtimes, due to the sparsity constraint. In case the timelimit is reached we take the best known solution provided by Gurobi.

5.2.1 The Knapsack Problem

In the following we show the results of Algorithm 1 and Algorithm 1 (Variant) on the KP instances described above. In Figure 4 the optimality gaps are shown in % for different values of k for the KP. The results show exactly the behaviour we analyzed in Section 3, namely the optimality gap decreases for increasing k and returns the optimal solution already for $k \geq 10$ for both instance sizes. For $k = 4$ the optimality gap is at most 1.5%. The results show that the optimality gaps increase for larger uncertainty sets (i.e. larger Γ) and for a larger dimension of

the problem. Furthermore Algorithm 1 (Variant) performs significantly better with only small increases in computation times as shown in Figure 5. Both algorithms are able to achieve small optimality gaps in less than 1 second.

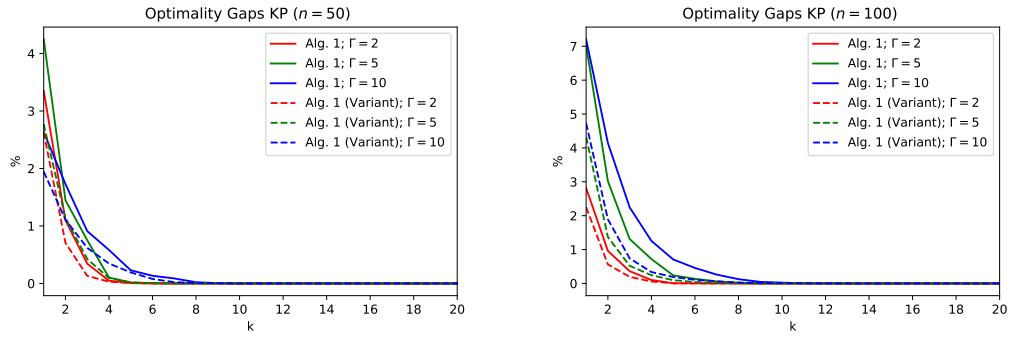

Figure 4: Average optimality gaps for KP instances with $n = 50$ (left) and $n = 100$ (right) for Algorithm 1 and Algorithm 1 (Variant) for different values of Γ.

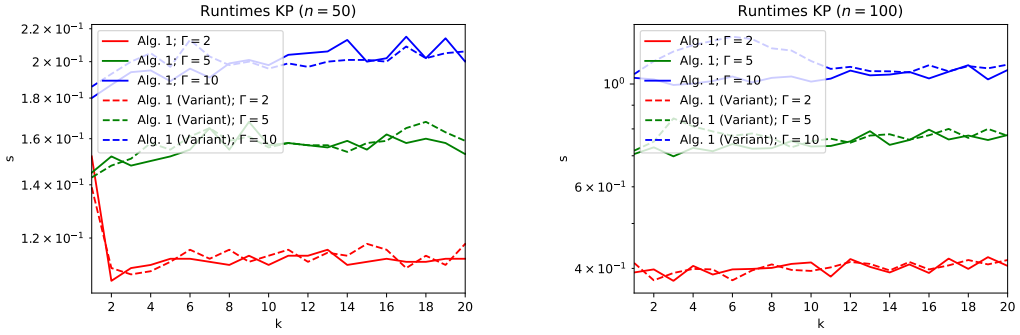

Figure 5: Average runtimes for KP instances with $n = 50$ (left) and $n = 100$ (right) for Algorithm 1 and Algorithm 1 (Variant) for different values of Γ.

5.2.2 Shortest Path Problem

In the following we show the results of Algorithm 1 and Algorithm 1 (Variant) on the SPP instances described above. In Figure 6 the optimality gaps are shown in % for different values of k for the SPP. Similar to the previous section, the results show exactly the behaviour we analyzed in Section 3, namely the optimality gap decreases for increasing k and returns the optimal solution already for $k \approx 20$ for both instance sizes. Compared to KP the optimality gaps are larger for small k. The results show that the optimality gaps increase for larger uncertainty sets (i.e. larger Γ) and for a larger dimension of the problem. Furthermore Algorithm 1 (Variant) performs significantly better. However this comes with larger computation times as shown in Figure 7. Interestingly the instances with midsize k are the hardest instances for Algorithm 1 (Variant) while for larger k the runtime decreases again. Possibly this is because the sparsity constraint is redundant for larger k. Algorithm 1 (Variant) is able to achieve optimality gaps below 10% for $k = 4$ and $\Gamma = 5$ in less than 10 seconds.

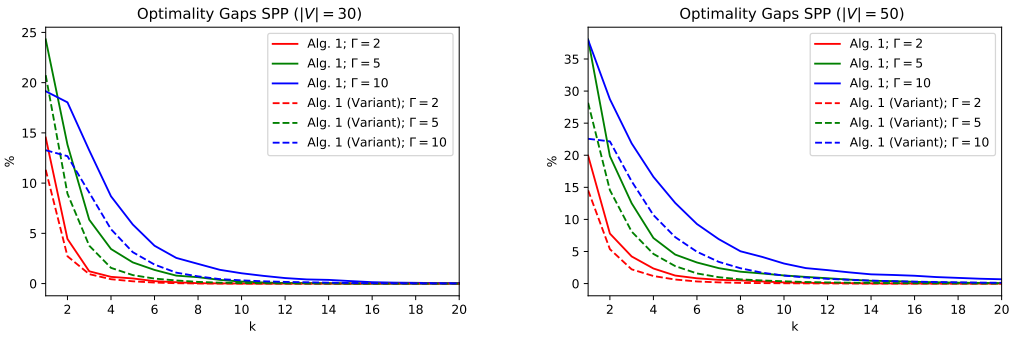

Figure 6: Average optimality gaps for SPP instances with 30 nodes (left) and 50 nodes (right) for Algorithm 1 and Algorithm 1 (Variant) for different values of Γ.

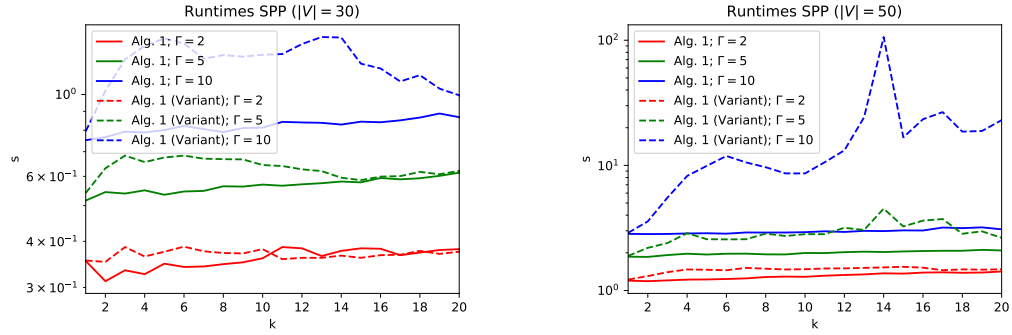

Figure 7: Average runtimes for SPP instances with 30 nodes (left) and 50 nodes (right) for Algorithm 1 and Algorithm 1 (Variant) for different values of Γ.

5.2.3 A Generative K-adaptability Problem

In the following we show the results of Algorithm 2 and Algorithm 2 (Variant) on the GP instances described above. In Figure 8 the optimality gaps are shown in % for different values of k. Similar to the previous section, the results show exactly the behaviour we analyzed in Section 3, namely the optimality gap decreases for increasing k. However since the first-stage solution calculated by Algorithm 2 is only an approximation, the optimality gap does not necessarily converge to zero for growing k as it was the case in the previous sections. The optimality gaps for small k are not larger than 5%. The results show that the optimality gaps increase for larger uncertainty sets (i.e. larger Γ) and for a larger dimension of the problem. Furthermore Algorithm 2 (Variant) performs significantly better. This comes with a small increase in computation times as shown in Figure 9. Similar to SPP the instances with midsize k are the hardest instances for Algorithm 1 (Variant) while for larger k the runtime decreases again. Possibly this is because the sparsity constraint is redundant for larger k. Algorithm 2 (Variant) is able to achieve optimality gaps below 2% for $k = 4$ in less than 1000 seconds.

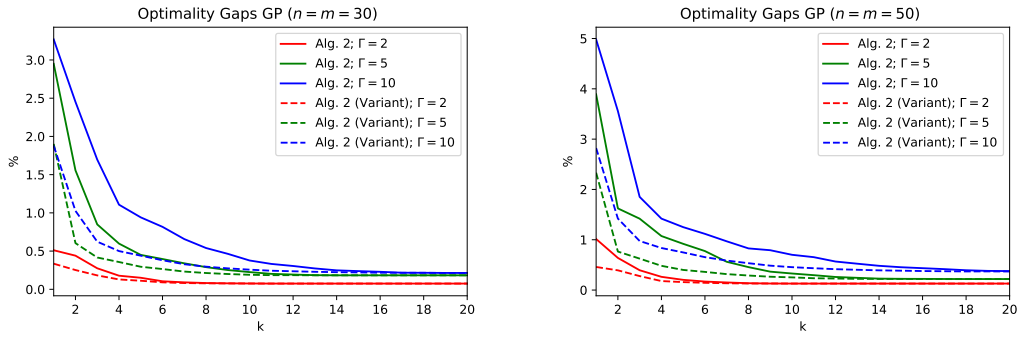

Figure 8: Average optimality gaps for GP instances with $n = m = 30$ (left) and $n = m = 50$ (right) for Algorithm 2 and Algorithm 2 (Variant) for different values of Γ.

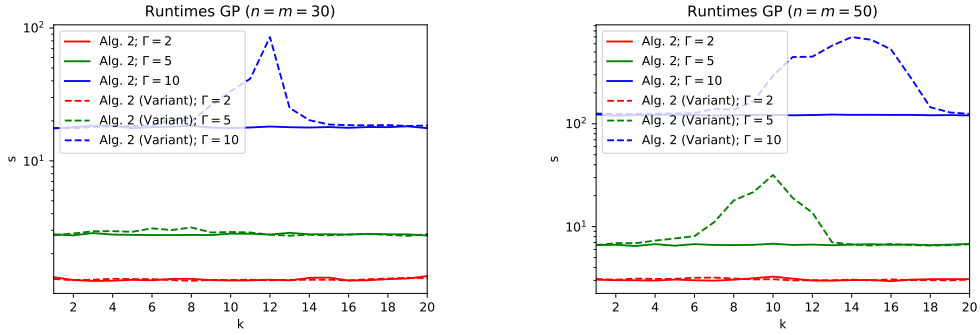

Figure 9: Average runtimes for GP instances with $n = m = 30$ (left) and $n = m = 50$ (right) for Algorithm 2 and Algorithm 2 (Variant) for different values of Γ.

5.2.4 Network Construction Shortest Path Problem

In the following we show the results of Algorithm 2 and Algorithm 2 (Variant) on the 2SP instances described above. In Figure 10 the optimality gaps are shown in % for different values of k. The optimality gap decreases for increasing k. However since the first-stage solution calculated by Algorithm 2 is only an approximation, the optimality gap does not necessarily converge to zero for growing k as it was the case for the min-max-min instances. The approximate first-stage solution is significantly worse compared to GP and the optimality gaps remain large. Here better problem-specific approximation algorithms for finding good two-stage solutions have to be developed. The results show that the optimality gaps increase for larger uncertainty sets (i.e. larger Γ) and for a larger dimension of the problem. Both, Algorithm 2 and Algorithm 2 (Variant) perform similarly in terms of optimality gap and runtimes. Interestingly, the runtimes for instances with $\Gamma = 10$ are smaller for Algorithm 2 (Variant). In total the runtime is always smaller than 600 seconds.

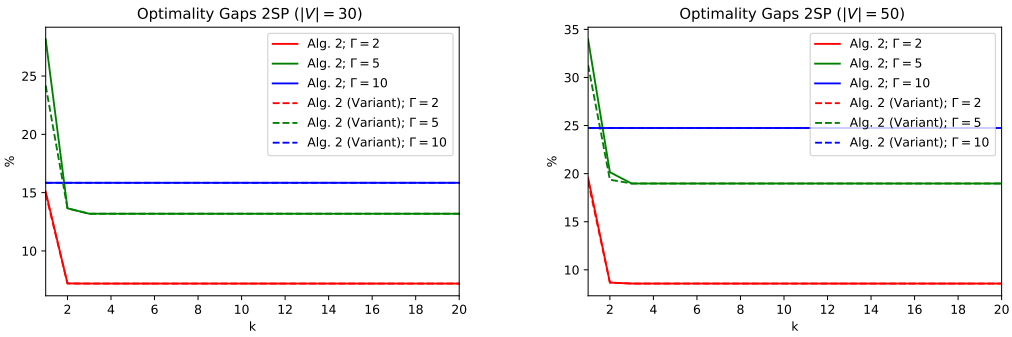

Figure 10: Average optimality gaps for 2SP instances with 30 nodes (left) and 50 nodes (right) for Algorithm 2 and Algorithm 2 (Variant) for different values of Γ.

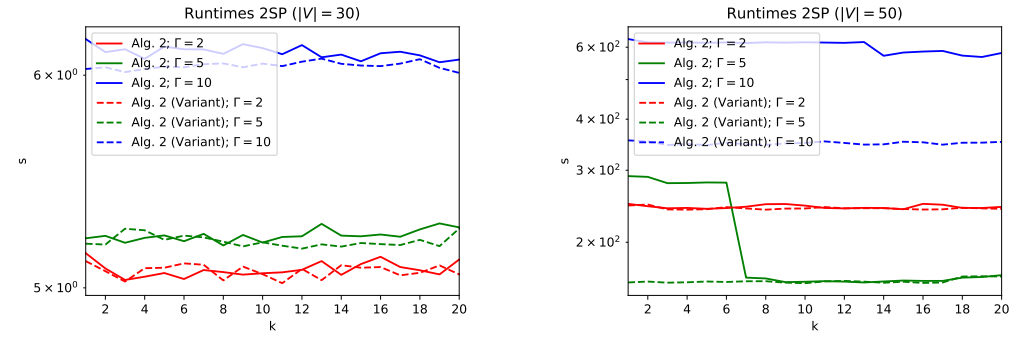

Figure 11: Average runtimes for 2SP instances with 30 nodes (left) and 50 nodes (right) for Algorithm 2 and Algorithm 2 (Variant) for different values of Γ.

6 Conclusion

In this work we study the min-max-min robust problem for binary optimization problems with uncertain costs. We focus on the case when the number of calculated solutions k is of intermediate size, i.e. either it is smaller but close to n or a fraction of n. We present an algorithm with provable additive and multiplicative approximation guarantees (involving problem-specific parameters) for both cases. The derived approximation guarantees depend on the size of the uncertainty set and the dimension of the problem and can be large for small k, but converge quickly to zero with increasing k. For special cases the guarantees provide an constant approximation guarantee (independent of n). Furthermore we calculate the range of parameter k for which a certain guarantee is achieved. We show that surprisingly if k is close to n, the min-max-min problem remains theoretically tractable and can be solved exactly and approximately in polynomial time for a large set of combinatorial problems if some problem parameters are fixed.

We extend the previous results to derive an approximation algorithm for k-adaptability problems and to derive lower bounds on the number of second-stage policies k which have to be used to achieve a certain approximation guarantee for the exact two-stage robust problem. We show that these bounds can also be used to approximate the recoverable robust problem.

Finally, the theoretical insights are confirmed by computations on min-max-min and k-adaptability instances from the literature. The experiments show that the presented algorithm is able to calculate solutions with small optimality gap in seconds, where the optimality gaps quickly go to zero with increasing k.

The results of this paper indicate that solving the min-max-min problem (and to some extend the k-adaptability problem) gets easier if k gets larger. This is in contrast to the results usually presented in the min-max-min and k-adaptability literature, where the number of solved instances (during a given timelimit) is often decreasing in k. However, if the approximation algorithms derived in this work (and the presented lower bounds) would be used to calculate solutions with good optimality guarantee, most of the known exact algorithms will scale much better for growing k due to the presented behaviour of the approximation bounds. Hence, we strongly recommend to use the presented approximation algorithms and lower bound calculations.

While the approximation quality in dependence of k for objective uncertainty is now well understood, for future work it would be interesting to derive similar results for the case that uncertain parameters are also allowed to appear in the constraints. However, the analysis presented in this work cannot be directly extended to this case and a completely new approach has to be developed.

References

[1] D. Adjiashvili, S. Stiller, and R. Zenklusen. Bulk-robust combinatorial optimization. *Mathematical Programming*, 149(1):361–390, 2015.

[2] 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.

[3] A. N. Arslan, M. Poss, and M. Silva. Min-sup-min robust combinatorial optimization with few recourse solutions. *INFORMS Journal on Computing*, 34(4):2212–2228, 2022.

[4] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton university press, 2009.

[5] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

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

[7] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[8] D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010.

[9] D. Bertsimas, V. Goyal, and X. A. Sun. A geometric characterization of the power of finite adaptability in multistage stochastic and adaptive optimization. *Mathematics of Operations Research*, 36(1):24–54, 2011.

[10] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Math. Program.*, 98(1-3):49–71, 2003.

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

[12] J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[13] M. Bruglieri, M. Ehrgott, H. W. Hamacher, and F. Maffioli. An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints. *Discrete Applied Mathematics*, 154(9):1344–1357, 2006.

[14] C. Buchheim and J. Kurtz. Min-max-min robustness: a new approach to combinatorial optimization under uncertainty based on multiple solutions. *Electronic Notes in Discrete Mathematics*, 52:45–52, 2016.

[15] C. Buchheim and J. Kurtz. Min–max–min robust combinatorial optimization. *Mathematical Programming*, 163(1):1–23, 2017.

[16] C. Buchheim and J. Kurtz. Complexity of min–max–min robustness for combinatorial optimization under discrete uncertainty. *Discrete Optimization*, 28:1–15, 2018.

[17] C. Buchheim and J. Kurtz. Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization*, 6(3):211–238, 2018.

[18] E. M. Cepolina. A methodology for defining building evacuation routes. *Civil engineering and environmental systems*, 22(1):29–47, 2005.

[19] A. Chassein and M. Goerigk. On the complexity of min–max–min robustness with two alternatives and budgeted uncertainty. *Discrete Applied Mathematics*, 2020.

[20] A. B. Chassein, M. Goerigk, J. Kurtz, and M. Poss. Faster algorithms for min-max-min robustness for combinatorial problems with budgeted uncertainty. *European Journal of Operational Research*, 279(2):308–319, 2019.

[21] M. Claus and M. Simmoteit. A note on σ2p-completeness of a robust binary linear program with binary uncertainty set. *Operations Research Letters*, 48(5):594–598, 2020.

[22] A. Crema. Min max min robust (relative) regret combinatorial optimization. *Mathematical Methods of Operations Research*, 92(2):249–283, 2020.

[23] L. Eufinger, J. Kurtz, C. Buchheim, and U. Clausen. A robust approach to the capacitated vehicle routing problem with uncertain costs. *INFORMS Journal on Optimization*, 2(2):79–95, 2020.

[24] M. Fischetti and M. Monaci. Light robustness. In *Robust and online large-scale optimization*, pages 61–84. Springer, 2009.

[25] A. Ghahtarani, A. Saif, A. Ghasemi, and E. Delage. A double-oracle, logic-based benders decomposition approach to solve the k-adaptability problem. *Computers & Operations Research*, 155:106243, 2023.

[26] M. Goerigk and M. Khosravi. Optimal scenario reduction for one-and two-stage robust optimization with discrete uncertainty in the objective. *European Journal of Operational Research*, 2023.

[27] M. Goerigk and J. Kurtz. Data-driven prediction of relevant scenarios for robust optimization. *arXiv preprint arXiv:2203.16642*, 2022.

[28] M. Goerigk, J. Kurtz, and M. Poss. Min–max–min robustness for combinatorial problems with discrete budgeted uncertainty. *Discrete Applied Mathematics*, 285:707–725, 2020.

[29] J. Goh and M. Sim. Distributionally robust optimization and its tractable approximations. *Operations research*, 58(4-part-1):902–917, 2010.

[30] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[31] Gurobi Optimization, LLC. Gurobi 10 optimizer reference manual, 2023.

[32] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

[33] S. Kedad-Sidhoum, A. Medvedev, and F. Meunier. On the continuity assumption of' finite adaptability in multistage linear optimization"by bertsimas and caramanis. *arXiv preprint arXiv:2305.05399*, 2023.

[34] P. Kouvelis and G. Yu. *Robust discrete optimization and its applications*, volume 14. Springer Science & Business Media, 2013.

[35] H. Lefebvre, E. Malaguti, and M. Monaci. Adaptive robust optimization with discrete uncertainty. *preprint*, 2022.

[36] C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. *Robust and online large-scale optimization: Models and techniques for transportation systems*, pages 1–27, 2009.

[37] G. Mishra, S. Mazumdar, and A. Pal. Improved algorithms for the evacuation route planning problem. In *Combinatorial Optimization and Applications*, pages 3–19. Springer, 2015.

[38] V. Pillac, P. Van Hentenryck, and C. Even. A conflict-based path-generation heuristic for evacuation planning. *Transportation Research Part B: Methodological*, 83:136–150, 2016.

[39] M. Rungta, G. J. Lim, and M. Baharnemati. Optimal egress time calculation and path generation for large evacuation networks. *Annals of Operations Research*, 201(1):403–421, 2012.

[40] M. Skutella and W. Welz. Route planning for robot systems. In *Operations research proceedings 2010*, pages 307–312. Springer, 2011.

[41] V. Stienen, J. Wagenaar, D. den Hertog, and H. Fleuren. Optimal depot locations for humanitarian logistics service providers using robust optimization. *Omega*, 104:102494, 2021.

[42] A. Subramanyam. A lagrangian dual method for two-stage robust optimization with binary uncertainties. *Optimization and Engineering*, 23(4):1831–1871, 2022.

[43] A. Subramanyam, C. E. Gounaris, and W. Wiesemann. K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation*, pages 193–224, 2020.

[44] W. Wiesemann, D. Kuhn, and M. Sim. Distributionally robust convex optimization. *Operations Research*, 62(6):1358–1376, 2014.

Appendix

Proof of Lemma 16

Proof. Let $x^*(k) \in X$, $y^*(k) = \sum_{i\in[k]} \lambda_i y^i$ be an optimal solution of Problem (k-adapt) with parameter k. We may assume without loss of generality that $\lambda_1 \geq \ldots \geq \lambda_k$. Define a solution $x(s)$, $y(s)$ of the s-adaptable problem by setting $x^*(s) = x^*(k)$ and

$$y(s) := \sum_{i\in[s-1]} \lambda_i y^i + \left(\sum_{i=s}^{k} \lambda_i\right) y^s. \tag{11}$$

The latter solution is clearly feasible for the s-adaptable problem and hence

$$\operatorname{adapt}(s) \leq \max_{\xi\in U} d^\top x(s) + \xi^\top y(s).$$

Furthermore let $\xi^*(s) \in \arg\max_{\xi\in U} d^\top x(s) + \xi^\top y(s)$. It follows

$$\begin{aligned}
\operatorname{adapt}(s) - \operatorname{adapt}(k) &\leq \max_{\xi\in U} d^\top x(s) + \xi^\top y(s) - (\max_{\xi\in U} d^\top x^*(k) + \xi^\top y^*(k)) \\
&\leq \xi^*(s)^\top \left(y(s) - y^*(k)\right) \\
&= \sum_{i=s+1}^{k} \lambda_i \xi^*(s)^\top \left(y^s - y^i\right) \\
&\leq M(n) \left(\sum_{i=s+1}^{k} \lambda_i\right)
\end{aligned}$$

where the second inequality holds since $\xi^*(s)$ is a subgradient of the function $g(y) = \max_{\xi\in U} d^\top x + \xi^\top y$ in $y(s)$ and since $x^*(k) = x(s)$. For the first equality we used the definition of $y(s)$ and $y^*(k)$ and for the last inequality we used the assumption $\underline{p}(n) \leq \|y\|_0 \leq \bar{p}(n)$ and $m_\infty \leq \xi^*(s)_j \leq M_\infty$ for all $j \in [n]$. Due to the sorting $\lambda_1 \geq \ldots \geq \lambda_k$ and since the sum over all λ_i is one, we have $\lambda_i \leq \frac{1}{i}$ and hence

$$\begin{aligned}
M(n) \left(\sum_{i=s+1}^{k} \lambda_i\right) &\leq M(n) \left(\sum_{i=s+1}^{k} \frac{1}{i}\right) \\
&\leq M(n) \frac{k-s}{s+1}
\end{aligned}$$

which proves the result. □