Constructing Representative Scenarios to Approximate Robust Combinatorial Optimization Problems

Marc Goerigk

Department of Management Science, Lancaster University, United Kingdom

Abstract

In robust combinatorial optimization with discrete uncertainty, two general approximation algorithms are frequently used, which are both based on constructing a single scenario representing the whole uncertainty set. In the midpoint method, one optimizes for the average case scenario. In the element-wise worst-case approach, one constructs a scenario by taking the worst case in each component over all scenarios. Both methods are known to be $N$-approximations, where $N$ is the number of scenarios.

In this paper, these results are refined by reconsidering their respective proofs as optimization problems. We present a linear program to construct a representative scenario for the uncertainty set, which guarantees an approximation guarantee that is at least as good as for the previous methods. Incidentally, we show that the element-wise worst-case approach can have an advantage over the midpoint approach if the number of scenarios is large. In numerical experiments on the selection problem we demonstrate that our approach can improve the approximation guarantee of the midpoint approach by around 20%.

Keywords: robust optimization; combinatorial optimization; approximation algorithms

1 Introduction

We consider combinatorial optimization problems of the general form

$$\min_{x \in X} cx$$

where $c \geq 0$ is a cost vector, and $X \subseteq \{0, 1\}^n$ is a set of feasible solutions. As real-world problems may suffer from uncertainty, robust counterparts to combinatorial
problems have been considered in the literature, see [ABV09, KZ16] for surveys on the topic. The resulting robust (or min-max) optimization problem is then of the form
\[
\min_{\mathbf{x} \in X} \max_{\mathbf{c} \in \mathcal{U}} \mathbf{c}^\top \mathbf{x}
\]
where \( \mathcal{U} \) contains all possible cost vectors \( \mathbf{c}^1, \ldots, \mathbf{c}^N \) against we wish to protect.

As robust combinatorial problems are usually NP-hard, approximation methods have been considered [ABV07]. Two such heuristics stand out in the literature, as they are easy to use and implement, and have been providing the best-known approximation guarantee for a wide range of problems. While this guarantee has been improved for specific problems, they are still the best-known general methods (see [CG17]). Both algorithms are based on constructing a single scenario that represents the whole uncertainty \( \mathcal{U} \). For the midpoint algorithm, we use \( \mathbf{c}^* \) with 
\[
\hat{c}_i = \frac{1}{N} \sum_{j \in [N]} c^j_i \quad \text{for all } i \in [n].
\]
For the element-wise worst-case algorithm, we use \( \mathbf{c}^* \) by using 
\[
\bar{c}_i = \max_{j \in [N]} c^j_i.
\]
Let us denote by \( \mathbf{x}(\mathbf{c}) \) a minimizer for the nominal problem with costs \( \mathbf{c} \), and set \( \hat{\mathbf{x}} := \mathbf{x}(\mathbf{c}^*) \) (the midpoint solution) and \( \bar{\mathbf{x}} := \mathbf{x}(\mathbf{c}) \) (the element-wise worst-case solution). The following results can be found in [ABV09].

**Theorem 1.** The midpoint solution \( \hat{\mathbf{x}} \) is an \( N \)-approximation for MinMax.

**Theorem 2.** The element-wise worst-case solution \( \bar{\mathbf{x}} \) is an \( N \)-approximation for MinMax.

Frequently, problems with "nice" structure (such as shortest path, spanning tree, selection, or assignment) have been considered in the literature, where it is possible to solve the nominal problem in polynomial time. In particular, this setting makes it possible to solve both of the above approaches in polynomial time by solving one specific scenario (i.e., finding \( \mathbf{x}(\mathbf{c}^*) \) or \( \mathbf{x}(\mathbf{c}) \)). This can then be used, e.g., as part of a branch and bound procedure for the (hard) robust problem.

Recently, data-driven robust optimization approaches have been investigated in the literature (see, e.g., [DG17, BGK18]). This paper has a similar research outlook by using the available data for better approximation guarantees, instead of ignoring structure that may be present. In a similar spirit, by analyzing the symmetry of an uncertainty set, [Con12] is able to derive improved approximation bounds for the related MinMax Regret problem with compact uncertainty sets.

The contributions of this paper are as follows. By re-examining the proofs for Theorems 1 and 2 we present a linear program (LP) to construct a scenario \( \mathbf{c}' \) that is "representative" for the uncertainty set \( \mathcal{U} \). We show that the resulting solution \( \mathbf{x}(\mathbf{c}') \) has an approximation guarantee that is at least as good as the guarantee for \( \hat{\mathbf{x}} \) and \( \bar{\mathbf{x}} \). We also compare the midpoint and element-wise worst-case approach in more detail and find that the latter can outperform the former if the number of scenarios is large. In numerical experiments, we compare the quality of upper and lower bounds of our approach with the midpoint method, and demonstrate that it is possible to find considerably smaller a-priori and a-posteriori gaps by solving a simple linear program.
2 Scenario construction based on the midpoint approach

Let $OPT$ be the optimal objective value of problem MinMax, and let $\mathbf{x}^*$ be any optimal solution. We make the following distinctions.

**Definition 3.** Let some scenario $\mathbf{c}$ (not necessarily in $U$) be given. Then

$$UB(\mathbf{c}) = \max_{i \in [N]} \mathbf{c}^i \mathbf{x}(\mathbf{c})$$

is an upper bound on $OPT$. If it is possible to compute a lower bound from $\mathbf{c}$, we denote this as $LB(\mathbf{c})$, and a bound on the ratio as

$$r(\mathbf{c}) \geq UB(\mathbf{c})/LB(\mathbf{c})$$

We call $r(\mathbf{c})$ an a-priori bound, if it does not require the computation of $\mathbf{x}(\mathbf{c})$ to find. Otherwise, we call it an a-posteriori bound.

The reason for this distinction is that calculation of $\mathbf{x}$ can be costly, if the nominal problem is not solvable in polynomial time.

As an example, the midpoint method uses $\hat{\mathbf{c}} := \frac{1}{N} \sum_{i \in [N]} \mathbf{c}^i$. It comes with an a-priori bound that is $N$, but by using $LB(\hat{\mathbf{c}}) = \hat{\mathbf{c}} \mathbf{x}(\hat{\mathbf{c}})$, we can calculate a stronger a-posteriori bound.

We now consider the problem of finding a better a-priori bound than $N$. To this end, note that Theorem 1 can be proven in the following way.

**Proof of Theorem 1.**

$$UB(\hat{\mathbf{c}}) = \max_{i \in [N]} \mathbf{c}^i \hat{\mathbf{x}} \leq N\hat{\mathbf{x}} \leq N\mathbf{x}^* \leq N \max_{i \in [N]} \mathbf{c}^i \mathbf{x}^* = N \cdot OPT$$

To mirror the steps of this proof, let us consider the following optimization problem:

$$\min_{t,\mathbf{c}} \quad (1)$$

s.t. $\quad \max_{i \in [N]} \mathbf{c}^i \mathbf{x}(\mathbf{c}) \leq t \cdot \mathbf{c} \mathbf{x}(\mathbf{c}) \tag{2}$

$\quad \mathbf{c} \mathbf{x}^* \leq \max_{i \in [N]} \mathbf{c}^i \mathbf{x}^* \tag{3}$

**Lemma 4.** Let $(t, \mathbf{c})$ be a feasible solution to problem (1–3). Then, $\mathbf{x}(\mathbf{c})$ is a $t$-approximation for MinMax.

**Proof.** Analogous to the proof of Theorem 1.

Note that Problem (1–3) cannot be solved directly, as both the optimal solution $\mathbf{x}^*$ and $\mathbf{x}(\mathbf{c})$ are unknown. To circumvent these two issues, we use different, sufficient constraints instead.
Lemma 5. Let \( c \) fulfil
\[
\sum_{j \in S} c_j \leq t \sum_{j \in S} c_j \quad \forall i \in [N], S \subseteq [n] : |S| = k
\]
for some value of \( t \), and constant \( k \) such that \( k \leq \sum_{j \in [n]} x_j \) for all \( x \in X \). Then, \((t, c)\) also fulfils (2).

Proof. Let \( X = \{ j \in [n] : x_j(c) = 1 \} \) and \( S = \{ S \subseteq [n] : |S| = k, S \subseteq X \} \). Then, the number of sets \( S \) in \( \mathcal{S} \) containing a specific item \( j \in X \) is the same for all \( j \). Let \( \ell \) be this number. By summing (4) over all \( S \in \mathcal{S} \), we find that
\[
\ell \sum_{j \in X} c_j \leq t \ell \sum_{j \in X} c_j \quad \forall i \in [N]
\]
and the claim follows. \( \square \)

Note that for constant \( k \), it is possible in polynomial time to check if \( k \leq \sum_{j \in [n]} x_j \) for all \( x \in X \). Also, the set \( \mathcal{S} \) contains polynomially many elements. As an example, for \( k = 1 \), Constraint (4) becomes
\[
c_j \leq tc_j \quad \forall i \in [N], j \in [n]
\]
and for \( k = 2 \), it becomes
\[
c_j + c_l \leq t(c_j + c_l) \quad \forall i \in [N], j, l \in [n], j \neq l
\]
In general, the constraints for some fixed \( k \) also imply the constraints for any larger \( k \). This means that the larger the value of \( k \), the larger is the set of feasible solutions to our optimization problem, and the better approximation guarantees we can get.

Lemma 6. Let \( c \) be in \( \text{conv}(\mathcal{U}) = \text{conv}\{c^1, \ldots, c^N\} \). Then, \( c \) fulfils (3).

Proof. Let \( c = \sum_{i \in [N]} \lambda_i c^i \) with \( \sum_{i \in [N]} \lambda_i = 1 \) and \( \lambda_i \geq 0 \) for all \( i \in [N] \). Then, for any \( x \in X \),
\[
 cx = \sum_{i \in [N]} \lambda_i c^i x \leq \sum_{i \in [N]} \lambda_i \max_{j \in [N]} c_j x = \max_{i \in [N]} c^i x
\]
\( \square \)

We now consider the following linear program:
\[
\begin{align*}
\text{max } & \quad t \\
\text{s.t. } & \quad t \sum_{j \in S} c^i_j \leq \sum_{j \in S} c_j \quad \forall i \in [N], S \subseteq [n] : |S| = k \\
& \quad c^i = \sum_{i \in [N]} \lambda_i c^i \\
& \quad \sum_{i \in [N]} \lambda_i = 1 \\
& \quad \lambda_i \geq 0 \quad \forall i \in [N]
\end{align*}
\]
Note that we replaced variable \( t \) in Problem (1-3) with \( 1/t \) to linearize terms.
Theorem 7. Let \((t^*, c^*)\) be an optimal solution to Problem \((5–9)\). Then, \(x(c^*)\) is a \(1/t^*\)-approximation for MinMax, and \(1/t^* \leq N\).

Proof. By Lemmas 5 and 6, \((1/t^*, c^*)\) is feasible for Problem \((1–3)\). Using Lemma 4, we therefore find that \(x(c^*)\) is a \(1/t^*\)-approximation for MinMax.

To see that \(1/t^* \leq N\), note that \((1/N, \hat{c})\) is a feasible solution to Problem \((5–9)\).

Once a solution \((t^*, c^*)\) has been computed, we have found an a-priori approximation guarantee. If we then compute \(x(c^*)\), we can derive a lower bound \(c' x(c^*)\), as \(c^* \in \text{conv}(U)\), and an upper bound by calculating the objective value of \(x(c^*)\) for MinMax. This way, a stronger a-posteriori guarantee is found.

Example 8. We illustrate our approach using a small selection problem as an example. Given four items, the task is to choose two of them that minimize the worst-case costs over three scenarios. The upper part of Table 1 shows the item costs in each scenario.

| item | 1 | 2 | 3 | 4 |
|------|---|---|---|---|
| \(c^1\) | 5 | 5 | 3 | 3 |
| \(c^2\) | 3 | 8 | 9 | 7 |
| \(c^3\) | 3 | 2 | 1 | 6 |
| \(\hat{c}\) | 3.67 | 5.00 | 4.33 | 5.33 |
| \(c'\) | 3.75 | 6.88 | 6.75 | 5.50 |
| \(c''\) | 3.00 | 8.00 | 9.00 | 7.00 |

Table 1: Example item costs, with midpoint scenario (\(\hat{c}\)), our LP-based scenario with \(k = 1\) (\(c'\)), and with \(k = 2\) (\(c''\)).

The midpoint scenario (i.e., the average in each item) is shown in the row below (\(\hat{c}\)). An optimal solution for this scenario is to pack items 1 and 3. This means that we have an a-priori approximation ratio of \(N = 3\), and can calculate a lower bound \(LB(\hat{c}) = \hat{c} x = 8\) and an upper bound \(UB(\hat{c}) = \max_{i \in [N]} c^i \hat{c} = 12\). Combining lower and upper bound, we find the stronger a-posteriori bound of 1.50.

Using our linear program \((5–9)\) with \(k = 1\), we construct the scenario given in the next row (\(c'\)) and find an a-priori guarantee of 1.33. For this scenario, an optimal solution is to take items 1 and 4. Accordingly, we find a lower bound of 9.25, an upper bound of 10, and an a-posteriori ratio of 1.08.

Finally, we also use our LP with \(k = 2\) to find the scenario \(c''\) and an a-priori guarantee of 1. This means that even before we have solved the problem, we already know that the resulting solution will be optimal. Indeed, we find that packing items 1 and 4 gives the optimal solution with objective value 10.

Note that we can also use the linear program \((5–9)\) to strengthen the approximation guarantee of the midpoint scenario \(\hat{c}\) without calculating \(x\), by only keeping \(t\) variable.

We conclude this section by introducing an alternative approach to calculate a-posteriori bounds, which cannot be used for a-priori bounds. To this end, note
that
$$\max_{\mathbf{c}\in \text{conv}(\mathcal{U})} \min_{\mathbf{x}\in \mathcal{X}} \mathbf{c}^\top \mathbf{x} \leq \min_{\mathbf{x}\in \mathcal{X}} \mathbf{c}^\top \mathbf{x} \leq \max_{i\in [n]} \mathbf{c}_i^\top \mathbf{x}^* \leq N \cdot \text{OPT}$$

If the nominal problem can be written as a linear program, it can be dualized to find a compact formulation for the max-min problem. As both $\hat{\mathbf{c}}$ and the optimal solution to problem (5–9) are in $\text{conv}(\mathcal{U})$, this approach will result in a lower bound which will be at least as good as the lower bounds of the other two approaches. This may not result in a better ratio between upper and lower bound, however. We will test this approach in the experimental section.

3 On the element-wise worst-case

We now focus on the element-wise worst-case scenario $\bar{\mathbf{c}}$ with $\bar{c}_i = \max_{j\in [N]} c_{ij}$. A proof for Theorem 2 is the following.

**Proof of Theorem 2**

\[
UB(\bar{\mathbf{c}}) = \max_{i\in [N]} c_i^\top \bar{\mathbf{x}} \overset{(i)}{\leq} \bar{\mathbf{x}}^\top \mathbf{x}^* \overset{(ii)}{\leq} N \max_{i\in [N]} c_i^\top \mathbf{x}^* = N \cdot \text{OPT}
\]

Accordingly, we can generalize this proof to an optimization problem by writing

\[
\begin{align*}
\min_{t, \mathbf{c}} & \quad t \\
\text{s.t.} & \quad \max_{i\in [N]} c_i^\top \mathbf{x} \leq \mathbf{c}^\top \mathbf{x} \\
& \quad \mathbf{c}^\top \mathbf{x}^* \leq t \max_{i\in [N]} c_i^\top \mathbf{x}^*
\end{align*}
\]

By substituting $\mathbf{c}' := \mathbf{c}/t$, Problem (10–12) becomes equivalent to Problem (1–3). Hence, we can apply the same techniques to transform this into a conservative linear program (5–9) as in the previous section. Note, however, that while $\hat{\mathbf{c}}$ is a feasible solution for this problem, this may not be the case for $\bar{\mathbf{c}}$.

Related to the MINMAX approach is MINMAX REGRET, where objective values are normalized by the optimal objective value in each scenario, i.e.,

\[
\min_{\mathbf{x}\in \mathcal{X}, i\in [N]} \max_{j\in [N]} (\mathbf{c}_i^\top \mathbf{x} - \mathbf{c}_j^\top \mathbf{x}(\mathbf{c}_i)) \quad (\text{MINMAX REGRET})
\]

The following result is also from [ABV09].

**Theorem 9.** The midpoint algorithm is an $N$-approximation for MINMAX REGRET; this does not hold for the element-wise worst-case algorithm.

In combination with Theorems 1 and 2, this means that there are no known problem classes where the element-wise worst-case solution gives a better performance guarantee than the midpoint solution. The midpoint solution has also been found to be the best-known general approximation algorithm for interval uncertainty problems [KZ06]. For these reasons, the midpoint solution has seen more attention in the research literature than the element-wise worst-case approach. However, in the following we show that if the number of scenarios is large, we element-wise worst-case approach can perform better than the midpoint approach, i.e., not only the size of the uncertainty set plays a role for approximability, but also the problem dimension.
Theorem 10. The element-wise worst-case algorithm is a $|X|$-approximation for MINMAX, where $|X| = \max_{x \in X} \sum_{j \in [n]} x_j$.

Proof. It holds that
\[
\max_{i \in [N]} \sum_{j \in [n]} c^i_j x_j \leq \sum_{j \in [n]} c_j x_j \leq \sum_{i \in [N]} \max_{j \in [n]} c^i_j x^*_j \\
\leq |X| \cdot \max_{j \in [n]} c_j x^*_j = |X| \cdot \max_{i \in [N]} \max_{j \in [n]} c^i_j x^*_j \leq |X| \cdot \sum_{j \in [n]} c^*_j x_j = |X| \cdot \text{OPT}
\]

Note that $|X| \leq n$. The approximation guarantees from Theorems 1 and 2 are tight, as the following two examples for robust shortest path problems demonstrate (see also [ABV09]).

(a) Hard instance for the midpoint solution.

(b) Hard instance for the element-wise worst-case solution.

Figure 1: Example instances for robust shortest path with two scenarios.

In Figure 1(a), the midpoint solution cannot distinguish between the upper edge and the lower edge. Hence, in this case, the $N$-approximation guarantee is tight with $N = 2$. In Figure 1(b), the element-wise worst-case solution cannot differentiate between the upper and the lower path. This instance is an example where the $N$-approximation guarantee is tight for this approach.

Note that the instance from Figure 1(a) can be extended by using more scenarios, preserving that the midpoint solution is an $N$-approximation, without additional edges. This is not the case for the element-wise worst-case scenario in Figure 1(b). To extend this instance to more scenarios, additional edges are required. This demonstrates that the midpoint solution is not a $|X|$-approximation, as shown for the element-wise worst-case approach.

4 Experiments

To test the quality of our LP-based scenario construction approach, we consider instances of the selection problem (see, e.g., [KZ16]). Here, $X = \{x \in \{0, 1\}^n : \sum_{j \in [n]} x_j = p\}$ for some integer parameter $p$. We generate item costs $c^i_j$ by sampling uniformly i.i.d. from $\{0, 1, \ldots, 100\}$. We use instances sizing from $n = 10$, $p = 3$ to
Table 2: Average a-priori bounds.

|  n  |  p  |  N   | Mid-1-Pre | Mid-2-Pre | Mid-3-Pre | LP-1-Pre | LP-2-Pre | LP-3-Pre |
|----|----|-----|----------|----------|----------|----------|----------|----------|
| 10 | 3  | 2   | 1.86     | 1.75     | 1.65     | 1.70     | 1.57     | 1.46     |
| 10 | 3  | 5   | 2.41     | 2.09     | 1.90     | 1.83     | 1.67     | 1.54     |
| 10 | 3  | 10  | 2.45     | 2.13     | 1.97     | 1.79     | 1.65     | 1.53     |
| 10 | 3  | 50  | 2.26     | 2.10     | 2.00     | 1.59     | 1.53     | 1.46     |
| 10 | 3  | 100 | 2.18     | 2.08     | 2.00     | 1.52     | 1.48     | 1.43     |
| 20 | 6  | 2   | 1.93     | 1.86     | 1.80     | 1.84     | 1.76     | 1.70     |
| 20 | 6  | 5   | 2.66     | 2.32     | 2.14     | 2.09     | 1.94     | 1.82     |
| 20 | 6  | 10  | 2.63     | 2.32     | 2.16     | 2.01     | 1.89     | 1.80     |
| 20 | 6  | 50  | 2.32     | 2.18     | 2.09     | 1.77     | 1.73     | 1.69     |
| 20 | 6  | 100 | 2.23     | 2.13     | 2.06     | 1.70     | 1.67     | 1.64     |
| 30 | 9  | 2   | 1.96     | 1.92     | 1.87     | 1.90     | 1.84     | 1.79     |
| 30 | 9  | 5   | 2.78     | 2.45     | 2.27     | 2.24     | 2.08     | 1.97     |
| 30 | 9  | 10  | 2.73     | 2.42     | 2.26     | 2.13     | 2.03     | 1.94     |
| 30 | 9  | 50  | 2.36     | 2.22     | 2.14     | 1.87     | 1.83     | 1.79     |
| 30 | 9  | 100 | 2.26     | 2.16     | 2.10     | 1.79     | 1.77     | 1.74     |

Table 2: Average a-priori bounds.

\[ n = 30, \ p = 9 \] and use \( N \in \{2, 5, 10, 50, 100\} \). For each parameter combination, we generate 1000 instances and average results.

Table 2 shows the a-priori bounds for the midpoint approach when using our linear program (5–9) for evaluation with \( k = 1, k = 2 \) and \( k = 3 \) (Mid-1-Pre, Mid-2-Pre, and Mid-3-Pre, respectively). We compare this to the a-priori bounds that are found when also optimizing over the scenario \( \mathbf{c} \) for \( k = 1, k = 2 \) and \( k = 3 \) (LP-1-Pre, LP-2-Pre, and LP-3-Pre, respectively). Note that overall, all guarantees are considerably smaller than \( N \). Furthermore, our approach is able to improve the bound of the midpoint algorithm. On average, the guarantee that the midpoint approach gives is more than 20% larger than our guarantee.

We contrast the a-priori bounds with a-posteriori bounds in Table 3, i.e., we calculate the solutions \( \mathbf{x}(\mathbf{c}) \) for the respective scenarios \( \mathbf{c} \) and the resulting ratio of upper and lower bound. On average, the bound provided by the midpoint solution is around 17% larger than the bound provided by our approach with \( k = 2 \) or \( k = 3 \). The max-min approach (denoted by MM) performs slightly better than our approach (Mid-Post is on average 19% larger than MM-Post), but this comes without an a-priori guarantee, at the cost of higher computational effort, and it is not always possible to compute as explained in Section 2.

Finally, we show more details on the a-posteriori bounds by providing both the upper and lower bounds in Tables 4 and 5. We find that our approach gives both better upper, and better lower bounds than the midpoint approach. While the MaxMin approach provides the best lower bounds, its upper bounds are often worse than for the midpoint solution.
| n  | p  | N  | Mid-Post | LP-1-Post | LP-2-Post | LP-3-Post | MM-Post |
|----|----|----|----------|-----------|-----------|-----------|---------|
| 10 | 3  | 2  | 1.30     | 1.24      | 1.22      | 1.21      | 1.24    |
| 10 | 3  | 5  | 1.57     | 1.35      | 1.30      | 1.32      | 1.29    |
| 10 | 3  | 10 | 1.66     | 1.39      | 1.34      | 1.36      | 1.34    |
| 10 | 3  | 50 | 1.82     | 1.37      | 1.36      | 1.38      | 1.37    |
| 10 | 3  | 100| 1.85     | 1.35      | 1.35      | 1.36      | 1.35    |
| 20 | 6  | 2  | 1.21     | 1.18      | 1.17      | 1.16      | 1.14    |
| 20 | 6  | 5  | 1.40     | 1.30      | 1.26      | 1.24      | 1.19    |
| 20 | 6  | 10 | 1.47     | 1.33      | 1.28      | 1.28      | 1.24    |
| 20 | 6  | 50 | 1.59     | 1.34      | 1.31      | 1.32      | 1.32    |
| 20 | 6  | 100| 1.63     | 1.33      | 1.31      | 1.32      | 1.32    |
| 30 | 9  | 2  | 1.17     | 1.16      | 1.15      | 1.14      | 1.10    |
| 30 | 9  | 5  | 1.32     | 1.26      | 1.21      | 1.20      | 1.14    |
| 30 | 9  | 10 | 1.38     | 1.30      | 1.26      | 1.25      | 1.19    |
| 30 | 9  | 50 | 1.48     | 1.30      | 1.28      | 1.28      | 1.28    |
| 30 | 9  | 100| 1.52     | 1.30      | 1.28      | 1.28      | 1.30    |

Table 3: Average a-posteriori bounds.

| n  | p  | N  | OPT  | Mid-UB | LP-1-UB | LP-2-UB | LP-3-UB | MM-UB |
|----|----|----|------|--------|---------|---------|---------|-------|
| 10 | 3  | 2  | 96.6 | 108.0  | 105.3   | 103.8   | 103.3   | 110.3 |
| 10 | 3  | 5  | 142.9| 169.5  | 162.8   | 158.0   | 158.8   | 165.9 |
| 10 | 3  | 10 | 170.4| 199.3  | 198.2   | 189.0   | 189.1   | 202.0 |
| 10 | 3  | 50 | 219.0| 248.3  | 249.8   | 241.9   | 239.9   | 254.1 |
| 10 | 3  | 100| 234.8| 260.4  | 262.6   | 256.3   | 253.6   | 265.4 |
| 20 | 6  | 2  | 172.1| 193.7  | 190.6   | 188.9   | 187.5   | 189.1 |
| 20 | 6  | 5  | 247.6| 296.6  | 289.4   | 282.2   | 280.2   | 276.9 |
| 20 | 6  | 10 | 292.7| 351.0  | 346.2   | 334.6   | 332.3   | 337.2 |
| 20 | 6  | 50 | 369.4| 431.8  | 438.8   | 424.6   | 420.6   | 440.3 |
| 20 | 6  | 100| 395.6| 457.7  | 461.2   | 450.8   | 446.6   | 464.5 |
| 30 | 9  | 2  | 247.2| 276.1  | 273.9   | 270.0   | 271.9   | 266.0 |
| 30 | 9  | 5  | 351.2| 416.2  | 408.6   | 398.3   | 395.9   | 384.1 |
| 30 | 9  | 10 | 409.2| 491.1  | 483.3   | 471.7   | 467.7   | 461.6 |
| 30 | 9  | 50 | 513.1| 605.5  | 610.3   | 592.4   | 588.6   | 607.0 |
| 30 | 9  | 100| 547.5| 638.3  | 645.1   | 628.5   | 623.6   | 648.3 |

Table 4: Average upper bounds.
Table 5: Average lower bounds.

| n  | p  | N  | OPT | Mid-LB | LP-1-LB | LP-2-LB | LP-3-LB | MM-LB |
|----|----|----|-----|--------|---------|---------|---------|-------|
| 10 | 3  | 2  | 96.6| 82.9   | 85.1    | 85.8    | 86.1    | 90.1  |
| 10 | 3  | 5  | 142.9| 108.3| 121.9  | 122.1    | 121.1   | 129.4 |
| 10 | 3  | 10 | 170.4| 120.3| 143.6  | 141.6    | 139.6   | 151.2 |
| 10 | 3  | 50 | 219.0| 136.7| 183.0  | 178.2    | 174.4   | 186.2 |
| 10 | 3  | 100| 234.8| 140.8| 194.8  | 190.6    | 186.2   | 196.6 |
| 20 | 6  | 2  | 172.1| 160.5| 161.6  | 162.1    | 162.4   | 166.2 |
| 20 | 6  | 5  | 247.6| 212.9| 223.7  | 225.2    | 225.6   | 234.1 |
| 20 | 6  | 10 | 292.7| 238.5| 260.9  | 261.2    | 260.3   | 272.9 |
| 20 | 6  | 50 | 369.4| 272.5| 327.8  | 323.6    | 319.9   | 333.1 |
| 20 | 6  | 100| 395.6| 280.8| 348.1  | 343.8    | 339.7   | 351.2 |
| 30 | 9  | 2  | 247.2| 236.3| 237.2  | 237.5    | 237.8   | 242.1 |
| 30 | 9  | 5  | 351.2| 316.3| 325.0  | 328.3    | 328.9   | 337.9 |
| 30 | 9  | 10 | 409.2| 355.1| 373.2  | 375.6    | 375.8   | 389.2 |
| 30 | 9  | 50 | 513.1| 408.0| 467.8  | 464.3    | 460.7   | 475.3 |
| 30 | 9  | 100| 547.5| 420.4| 495.9  | 491.5    | 487.4   | 500.1 |

5 Conclusion

Most robust combinatorial optimization problems are hard, which has lead to the development of general approximation algorithms. The two best-known such approaches are the midpoint method and the element-wise worst-case approach. Both rely on creating a single scenario that is representative for the whole uncertainty set. By reconsidering the respective proofs that both are $N$-approximation algorithms, we find an optimization problem to construct a representative scenario that results in an approximation which is at least as good as for the previous two scenarios.

In computational experiments using the selection problem, we test this approach numerically. We find that the midpoint method gives a guarantee that is about 20% larger than ours, while we only need to solve a simple linear program to construct the representative scenario. The improved a-priori guarantee is also reflected in an improved a-posteriori guarantee, with our approach providing both better upper and lower bounds than before. This smaller gap could potentially be used within branch-and-bound algorithms for a more efficient search for an optimal solution.

References

[ABV07] H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation of min–max and min–max regret versions of some combinatorial optimization problems. European Journal of Operational Research, 179(2):281 – 290, 2007.

[ABV09] H. Aissi, C. Bazgan, and D. Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. European Journal of Operational Research, 197(2):427 – 438, 2009.

[BGK18] D. Bertsimas, V. Gupta, and N. Kallus. Data-driven robust optimization. Mathematical Programming, 167(2):235–292, 2018.
A. Chassein and M. Goerigk. On scenario aggregation to approximate robust optimization problems. *Optimization Letters*, https://doi.org/10.1007/s11590-017-1206-x, 2017. Available online, to appear.

E. Conde. On a constant factor approximation for minmax regret problems using a symmetry point scenario. *European Journal of Operational Research*, 219(2):452–457, 2012.

T. Dokka and M. Goerigk. An Experimental Comparison of Uncertainty Sets for Robust Shortest Path Problems. In Gianlorenzo D’Angelo and Twan Dollevoet, editors, *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*, volume 59 of *OpenAccess Series in Informatics (OASIcs)*, pages 16:1–16:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

A. Kasperski and P. Zieliński. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97(5):177–180, 2006.

A. Kasperski and P. Zieliński. Robust discrete optimization under discrete and interval uncertainty: A survey. In *Robustness Analysis in Decision Aiding, Optimization, and Analytics*, pages 113–143. Springer, 2016.