Combinatorial two-stage minmax regret problems under interval uncertainty

Marc Goerigk¹ · Adam Kasperski² · Paweł Zieliński²

Accepted: 3 November 2020 / Published online: 24 November 2020
© The Author(s) 2020

Abstract

In this paper a class of combinatorial optimization problems is discussed. It is assumed that a feasible solution can be constructed in two stages. In the first stage the objective function costs are known while in the second stage they are uncertain and belong to an interval uncertainty set. In order to choose a solution, the minmax regret criterion is used. Some general properties of the problem are established and results for two particular problems, namely the shortest path and the selection problem, are shown.

Keywords Robust optimization · Combinatorial optimization · Minmax regret · Two-stage optimization · Complexity

1 Introduction

Consider the following deterministic single-stage *combinatorial optimization problem*:

$$\text{opt}^{\mathcal{P}}(\mathbf{c}) = \min_{\mathbf{x} \in \mathcal{X}} \mathbf{c}^T \mathbf{x}, \quad (\mathcal{P})$$

where $\mathcal{X} \subseteq \{0, 1\}^n$ is a set of *feasible solutions* and $\mathbf{c} \in \mathbb{R}_+^n$ is a vector of nonnegative objective function costs. Typically, $\mathcal{X}$ is described by a system of linear constraints involving binary variables x_i , $i \in [n]$ (we will use the notation $[n] = \{1, \dots, n\}$), which leads to a 0-1 programming problem. Solution $\mathbf{x} \in \mathcal{X}$ can be interpreted as a characteristic vector of some finite element set E . For example, E can be a set of edges of a graph $G = (V, E)$ and $\mathbf{x}$ describes some objects in G such as paths, trees, matchings etc.

✉ Adam Kasperski
adam.kasperski@pwr.edu.pl

Marc Goerigk
marc.goerigk@uni-siegen.de

Paweł Zieliński
pawel.zielinski@pwr.edu.pl

¹ Network and Data Science Management, University of Siegen, Siegen, Germany

² Wrocław University of Science and Technology, Wrocław, Poland

In many practical applications the vector of objective function costs $\mathbf{c}$ is uncertain and it is only known to belong to an uncertainty set $\mathcal{U}$. There are various methods of defining $\mathcal{U}$ which depend on the application and the information available. Among the easiest and most common is the *interval uncertainty representation* (see, e.g., Kouvelis and Yu 1997), in which $c_i \in [\underline{c}_i, \bar{c}_i]$ for each $i \in [n]$ and $\mathcal{U} = \prod_{i \in [n]} [\underline{c}_i, \bar{c}_i] \subseteq \mathbb{R}_+^n$. In order to choose a solution, for a specified $\mathcal{U}$, one can apply a robust decision criterion, which takes into account the worst cost realizations. Under the interval uncertainty representation, the *minmax regret* criterion (also called *Savage criterion* (Savage 1951)) has attracted a considerable attention in the literature. The *regret* of a given solution $\mathbf{x} \in \mathcal{X}$ under a cost scenario $\mathbf{c} \in \mathcal{U}$ is the quantity $\mathbf{c}^T \mathbf{x} - \text{opt}^{\mathcal{P}}(\mathbf{c})$. It expresses a deviation of solution $\mathbf{x}$ from the optimum and can be interpreted as the maximal opportunity loss after $\mathbf{x}$ is implemented. In the *single-stage minmax regret* version of $\mathcal{P}$ we seek a solution minimizing the maximum regret, i.e. we study the following problem:

$$\min_{\mathbf{x} \in \mathcal{X}} \max_{\mathbf{c} \in \mathcal{U}} (\mathbf{c}^T \mathbf{x} - \text{opt}^{\mathcal{P}}(\mathbf{c})). \quad (\text{SSTR } \mathcal{P})$$

The SSTR $\mathcal{P}$ problem has been discussed in a number of papers, for example when $\mathcal{P}$ is the minimum spanning tree (Montemanni 2006; Montemanni and Gambardella 2005; Yaman et al. 2001), the shortest path (Karaşan et al. 2001; Montemanni and Gambardella 2004; Montemanni et al. 2004), the minimum s - t cut (Aissi et al. 2008; Kasperski and Zieliński 2007), the minimum assignment (Aissi et al. 2005; Pereira and Averbakh 2011), or the selection (Averbakh 2001; Conde 2004) problem (in this case $\mathcal{X} = \{\mathbf{x} \in \{0, 1\}^n : x_1 + \dots + x_n = p\}$ for some fixed $p \in [n]$). Surveys of known results in this area can be found in Aissi et al. (2009) and Kasperski and Zieliński (2016). Unfortunately, SSTR $\mathcal{P}$ turned out to be NP-hard for all previously mentioned problems (Aron and van Hentenryck 2004; Averbakh and Lebedev 2004; Zieliński 2004; Kasperski and Zieliński 2006b), with a notable exception when $\mathcal{P}$ is the selection problem, for which polynomial algorithms were established in Averbakh (2001) and Conde (2004). The SSTR $\mathcal{P}$ problem has some well known general properties. There is a nice characterization of scenario $\mathbf{c} \in \mathcal{U}$ maximizing the regret of a given solution $\mathbf{x}$ (called a *worst-case scenario* for $\mathbf{x}$), namely, $c_i = \bar{c}_i$ if $x_i = 1$ and $c_i = \underline{c}_i$ if $x_i = 0$ for each $i \in [n]$. Notice that this scenario depends only on $\mathbf{x}$ and the problem of computing the maximum regret of a given solution has the same complexity as $\mathcal{P}$. Also, there is a general 2-approximation algorithm known for SSTR $\mathcal{P}$, under the assumption that $\mathcal{P}$ is polynomially solvable (Kasperski and Zieliński 2006a; Conde 2010; Chassein and Goerigk 2015). We get a 2-approximate solution by solving $\mathcal{P}$ under the so-called *midpoint scenario* $\mathbf{c}^m \in \mathcal{U}$ such that $c_i^m = (\underline{c}_i + \bar{c}_i)/2$ for each $i \in [n]$. Exact algorithms for solving SSTR $\mathcal{P}$ are based on compact mixed-integer programming (MIP) formulations (see, e.g., Yaman et al. 2001), when $\mathcal{P}$ has a special structure, or constraint generation technique in general (see, e.g., Montemanni 2006; Pereira and Averbakh 2013).

In some applications a solution from $\mathcal{X}$ can be constructed in two stages. Namely, a partial solution is chosen now (in the first stage) and is completed in the future (in the second stage). The current, first-stage costs are known while the future second-stage costs are uncertain and belong to an uncertainty set $\mathcal{U}$. However, the partial solution can be completed after a second-stage cost scenario is revealed. The problem consists in computing a best first-stage solution, which corresponds to the decision which must be made now. The two-stage approach has a long tradition in stochastic optimization (see, e.g., Kall and Mayer 2005). When a probability distribution in $\mathcal{U}$ is unknown, then a robust two-stage version of $\mathcal{P}$ can be considered. First such a model was discussed in Katriel et al. (2008) for the assignment problem. This approach was also applied to the minimum spanning tree (Kasperski and Zieliński 2011) and the

selection problems (Chassein et al. 2018). In these papers the robust minmax criterion has been applied, i.e. a first-stage solution is determined minimizing the largest total first and second-stage cost. Two-stage problems with the maximum regret criterion have recently been considered in Poursoltani and Delage (2019) with a focus on continuous problems, and in Crema (2020) under the min-max-min setting.

In this paper we wish to investigate the two-stage version of problem $\mathcal{P}$ under the interval uncertainty representation. Namely, for each second-stage cost an interval of its possible values is provided. We use the minmax regret criterion to choose a solution. The interpretation of this problem is the same as in the case of SSTR $\mathcal{P}$. We seek a first-stage solution, which minimizes the maximum regret, i.e. the maximum distance to a best first-stage solution. We will show that this problem has different properties than its single-stage counterpart. In particular, there is no easy characterization of a worst-case scenario of a given first-stage solution, although there is still a worst-case scenario which is extreme (the second-stage costs take their upper or lower bounds under this scenario). In fact, the problem of computing the maximum regret can be NP-hard, even if $\mathcal{P}$ is solvable in polynomial time. Also, the midpoint heuristic does not guarantee any approximation ratio in general. We will show a general method of solving the problem, which is based on a MIP formulation. We then study two special cases, when $\mathcal{P}$ is the shortest path and the selection problem.

This paper is organized as follows. In Sect. 2 we state the problem. We also consider three inner problems, in particular the problem of computing the maximum regret of a given first-stage solution. In Sect. 3, we construct MIP formulations, which can be used to compute exact or approximate solutions. Section 4 is devoted to the two-stage version of the shortest path problem. We proceed with the study of two variants of this problem, which have different computational properties. We show that both computing an optimal first-stage solution and the maximum regret of a given first-stage solution are NP-hard. In Sect. 5 we discuss the selection problem. We show that for this problem the maximum regret of a given first-stage solution can be computed in polynomial time and the optimal first-stage solution can be determined by using a compact MIP formulation. We also propose a greedy heuristic for this problem. Numerical results are presented in Sect. 6. Finally, the paper is concluded and further research questions are pointed out in Sect. 7.

2 Problem formulation

In this paper we assume that a solution from $\mathcal{X}$ can be built in two-stages. Given a vector $\mathbf{x} \in \{0, 1\}^n$, let

$$\mathcal{R}(\mathbf{x}) = \{\mathbf{y} \in \{0, 1\}^n : \mathbf{x} + \mathbf{y} \in \mathcal{X}\} \quad (1)$$

be the set of *recourse actions* for $\mathbf{x}$. Vector $\mathbf{y} \in \mathcal{R}(\mathbf{x})$ is a completion of the partial solution $\mathbf{x}$ to a feasible one. Let

$$\mathcal{X}' = \{\mathbf{x} \in \{0, 1\}^n : \mathcal{R}(\mathbf{x}) \neq \emptyset\}$$

be the set of *feasible first-stage solutions*. Define $\mathcal{Z} = \{(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^{2n} : \mathbf{x} \in \mathcal{X}', \mathbf{y} \in \mathcal{R}(\mathbf{x})\}$ as the set of all possible combinations between partial first-stage solutions and recourse actions. Given a first-stage cost vector $\mathbf{C} \in \mathbb{R}_+^n$ and a second-stage cost vector $\mathbf{c} \in \mathbb{R}_+^n$, we consider the following *two-stage problem*:

$$\text{Opt}(\mathbf{c}) = \min_{(\mathbf{x}, \mathbf{y}) \in \mathcal{Z}} (\mathbf{C}^T \mathbf{x} + \mathbf{c}^T \mathbf{y}). \quad (\text{TST } \mathcal{P})$$

Given $\mathbf{x} \in \mathcal{X}'$ and $\mathbf{c} \in \mathbb{R}_+^n$, we will also examine the following *incremental problem*:

$$\text{Inc}(\mathbf{x}, \mathbf{c}) = \mathbf{C}^T \mathbf{x} + \min_{\mathbf{y} \in \mathcal{R}(\mathbf{x})} \mathbf{c}^T \mathbf{y} \quad (\text{INC } \mathcal{P})$$

in which we seek a best recourse action for $\mathbf{x} \in \mathcal{X}'$ and $\mathbf{c}$. The quantity $\text{Inc}(\mathbf{x}, \mathbf{c}) - \text{Opt}(\mathbf{c})$ is called the *regret* of $\mathbf{x}$ under $\mathbf{c}$. Suppose that the second-stage costs are uncertain and we only know that $c_i \in [\underline{c}_i, \bar{c}_i]$ for each $i \in [n]$. We thus consider the *interval uncertainty representation* $\mathcal{U} = \prod_{i \in [n]} [\underline{c}_i, \bar{c}_i]$. Each possible second-stage cost vector $\mathbf{c} \in \mathcal{U}$ is called a *scenario*. Let us define the maximum regret of a given first-stage solution $\mathbf{x} \in \mathcal{X}'$ as follows:

$$Z(\mathbf{x}) = \max_{\mathbf{c} \in \mathcal{U}} (\text{Inc}(\mathbf{x}, \mathbf{c}) - \text{Opt}(\mathbf{c})). \quad (2)$$

A scenario which maximizes the right hand side of (2) is called a *worst-case scenario* for $\mathbf{x}$. In this paper we study the following *two-stage minmax regret problem*:

$$\min_{\mathbf{x} \in \mathcal{X}'} Z(\mathbf{x}) = \min_{\mathbf{x} \in \mathcal{X}'} \max_{\mathbf{c} \in \mathcal{U}} \min_{\mathbf{y} \in \mathcal{R}(\mathbf{x})} (\mathbf{C}^T \mathbf{x} + \mathbf{c}^T \mathbf{y} - \text{Opt}(\mathbf{c})). \quad (\text{TSTR } \mathcal{P})$$

Let us illustrate TSTR $\mathcal{P}$ when $\mathcal{P}$ is the SHORTEST PATH problem shown in Fig. 1a. In this case, $\mathcal{X}$ is the set of characteristic vectors of the simple $s - t$ paths in a given network $G = (V, A)$.

Let $\mathbf{x} = (x_{s1}, x_{s2}, x_{12}, x_{1t}, x_{2t})^T \in \mathcal{X}'$ denote a first-stage solution to the instance in Fig. 1a. A candidate solution is $\mathbf{x}' = (1, 0, 0, 0, 0)^T$, in which the arc $(s, 1)$ is selected in the first stage. Under scenario $\underline{\mathbf{c}} = (0, 0, 0, 0, 0)^T$, this partial solution can be completed to a path by choosing arc $(1, t)$ with total costs $1 + 0 = 1$. As $\text{Opt}(\underline{\mathbf{c}}) = 0$, the maximum regret of $\mathbf{x}'$ is at least 1. It can be verified that there is no other scenario that results in a higher regret. As a second example, the first-stage solution $(0, 1, 0, 0, 0)^T$ has the maximum regret equal to 10. Indeed in a worst scenario $\mathbf{c}'$ the cost of $(2, t)$ is set to 10 and the costs of the remaining arcs are set to 0. The arcs $(2, t)$ must be selected in the second stage and $\text{opt}(\mathbf{c}') = 0$, which results in the regret of 10. Notice also that solution $(0, 0, 0, 0, 0)^T$, when no arc is selected in the first stage, has the maximum regret equal to 2, achieved by using scenario $\bar{\mathbf{c}} = (2, 1, 0, 6, 10)$. A full enumeration reveals that $\mathbf{x}'$ is in fact optimal.

The instance in Fig. 1b demonstrates that the mid-point heuristic does not guarantee the approximation ratio of 2 for TSTR $\mathcal{P}$. Indeed, if we solve the TST SHORTEST PATH problem for the second-stage midpoint scenario $\mathbf{c}^m = (b/2, a)^T$, then we get solution $\mathbf{x}^m = (0, 1)^T$ with $Z(\mathbf{x}^m) = a - \epsilon$. But the optimal first-stage solution is $\mathbf{x} = (0, 0)^T$ with $Z(\mathbf{x}) = \epsilon$. Hence the ratio $Z(\mathbf{x}^m)/Z(\mathbf{x}) = (a - \epsilon)/\epsilon$ can be arbitrarily large.

Fig. 1 Two instances of TSTR SHORTEST PATH. In the instance b) we assume that $b \gg 2a \gg \epsilon > 0$

3 Mixed integer programming formulations

In this section we construct mixed integer programming formulations for computing the maximum regret $Z(\mathbf{x})$ of a given first-stage solution $\mathbf{x} \in \mathcal{X}'$ and solving the TSTR $\mathcal{P}$ problem. In particular, we will show that for each $\mathbf{x} \in \mathcal{X}'$, there exists a worst-case scenario which is extreme, i.e. which belongs to $\prod_{i \in [n]} \{\underline{c}_i, \bar{c}_i\}$. In the following, we will use $\underline{\mathbf{c}}$ to denote the scenario $(\underline{c}_1, \dots, \underline{c}_n)^T$. Fix $(\mathbf{u}, \mathbf{v}) \in \mathcal{Z}$ and define

$$Z_{(\mathbf{u}, \mathbf{v})}(\mathbf{x}) = \mathbf{C}^T \mathbf{x} - \mathbf{C}^T \mathbf{u} + \max_{\mathbf{c} \in \mathcal{U}} \min_{\mathbf{y} \in \mathcal{R}(\mathbf{x})} \mathbf{c}^T (\mathbf{y} - \mathbf{v}).$$

It is easy to verify that

$$Z(\mathbf{x}) = \max_{(\mathbf{u}, \mathbf{v}) \in \mathcal{Z}} Z_{(\mathbf{u}, \mathbf{v})}(\mathbf{x}). \quad (3)$$

Proposition 1 *It holds that*

$$Z_{(\mathbf{u}, \mathbf{v})}(\mathbf{x}) = \mathbf{C}^T \mathbf{x} - \mathbf{C}^T \mathbf{u} - \underline{\mathbf{c}}^T \mathbf{v} + \min_{\mathbf{y} \in \mathcal{R}(\mathbf{x})} \mathbf{c}_{\mathbf{v}}^T \mathbf{y}, \quad (4)$$

where scenario $\mathbf{c}_{\mathbf{v}} \in \mathcal{U}$ is such that $c_{vi} = \underline{c}_i$ if $v_i = 1$ and $c_{vi} = \bar{c}_i$ if $v_i = 0$.

Proof Write $Z_{(\mathbf{u}, \mathbf{v})}(\mathbf{x}) = \mathbf{C}^T \mathbf{x} - \mathbf{C}^T \mathbf{u} + t^*$, where t^* is computed by solving the following problem:

$$\max t \quad (5)$$

$$\text{s.t. } t \leq \mathbf{c}^T (\mathbf{y} - \mathbf{v}) \quad \forall \mathbf{y} \in \mathcal{R}(\mathbf{x}) \quad (6)$$

$$c_i \in [\underline{c}_i, \bar{c}_i] \quad \forall i \in [n] \quad (7)$$

An optimal solution to (6) and (7) can be determined as follows: for each $i \in [n]$, if $v_i = 1$, then $c_i = \underline{c}_i$ (because $y_i - v_i \leq 0$) and if $v_i = 0$, then $c_i = \bar{c}_i$ (because $y_i - v_i \geq 0$). This yields the scenario $\mathbf{c}_{\mathbf{v}}$. Observe also that $\mathbf{c}_{\mathbf{v}}^T (\mathbf{y} - \mathbf{v}) = -\underline{\mathbf{c}}^T \mathbf{v} + \min_{\mathbf{y} \in \mathcal{R}(\mathbf{x})} \mathbf{c}_{\mathbf{v}}^T \mathbf{y}$, which completes the proof. $\square$

Proposition 1 implies the following corollary:

Corollary 1 *For each $\mathbf{x} \in \mathcal{X}'$, there is a worst-case extreme scenario $\mathbf{c} \in \prod_{i \in [n]} \{\underline{c}_i, \bar{c}_i\}$.*

The result stated in Corollary 1 is analogous to the known result for the single-stage SSTR $\mathcal{P}$ problem (see, e.g., Aissi et al. 2009). However, in the two-stage model the worst-case scenario for $\mathbf{x} \in \mathcal{X}'$ is not completely characterized by $\mathbf{x}$. In order to compute $Z(\mathbf{x})$ one needs to find $(\mathbf{u}, \mathbf{v}) \in \mathcal{Z}$ maximizing the right-hand side of (3). In Sect. 4 we will show that the problem of computing $Z(\mathbf{x})$ is NP-hard, when $\mathcal{P}$ is the SHORTEST PATH problem. Using equality (4) from Proposition 1, we can compute the maximum regret of $\mathbf{x}$ in the following way:

$$\begin{aligned} Z(\mathbf{x}) = \max \quad & \mathbf{C}^T \mathbf{x} - \mathbf{C}^T \mathbf{u} - \underline{\mathbf{c}}^T \mathbf{v} + z \\ \text{s.t. } \quad & z \leq \mathbf{c}_{\mathbf{v}}^T \mathbf{y} \quad \forall \mathbf{y} \in \mathcal{R}(\mathbf{x}) \\ & (\mathbf{u}, \mathbf{v}) \in \mathcal{Z} \end{aligned}$$

which, by using the definition of $\mathbf{c}_{\mathbf{v}}$, can be stated equivalently as

$$\max \mathbf{C}^T \mathbf{x} - \mathbf{C}^T \mathbf{u} - \underline{\mathbf{c}}^T \mathbf{v} + z \quad (8)$$

$$\text{s.t. } z \leq \sum_{i \in [n]} (\underline{c}_i v_i + \bar{c}_i (1 - v_i)) y_i \quad \forall \mathbf{y} \in \mathcal{R}(\mathbf{x}) \quad (9)$$

$$(\mathbf{u}, \mathbf{v}) \in \mathcal{Z} \quad (10)$$

The number of constraints in (8)–(10) can be exponential in n . This problem can be solved by standard row generation techniques. If the problem of optimizing a linear objective function over $\mathcal{R}(\mathbf{x})$ can be written as a linear program, then it is also possible to find a compact reformulation of constraints (9) using primal–dual relationships (see, e.g., Papadimitriou and Steiglitz 1998). One such case will be demonstrated in Sect. 5. Let us now turn to the TSTR $\mathcal{P}$ problem. Again, using equality (3) we can express this problem as the following program:

$$\min z \quad (11)$$

$$\text{s.t. } z \geq Z_{(\mathbf{u}, \mathbf{v})}(\mathbf{x}) \quad \forall (\mathbf{u}, \mathbf{v}) \in \mathcal{Z} \quad (12)$$

$$\mathbf{x} \in \mathcal{X}' \quad (13)$$

Using Proposition 1, we can convert this model to

$$\min \mathbf{C}^T \mathbf{x} + z$$

$$\text{s.t. } z \geq \mathbf{c}_v^T \mathbf{y}^v - \mathbf{C}^T \mathbf{u} - \underline{\mathbf{c}}^T \mathbf{v} \quad \forall (\mathbf{u}, \mathbf{v}) \in \mathcal{Z}$$

$$\mathbf{y}^v \in \mathcal{R}(\mathbf{x}) \quad \forall (\mathbf{u}, \mathbf{v}) \in \mathcal{Z}$$

$$\mathbf{x} \in \mathcal{X}'$$

Finally, making use of the definition of $\mathbf{c}_v$ and $\mathcal{R}(\mathbf{x})$, we get the following MIP formulation for TSTR $\mathcal{P}$:

$$\min \mathbf{C}^T \mathbf{x} + z \quad (14)$$

$$\text{s.t. } z \geq \sum_{i \in [n]} (\underline{c}_i v_i + \bar{c}_i (1 - v_i)) y_i^v - \mathbf{C}^T \mathbf{u} - \underline{\mathbf{c}}^T \mathbf{v} \quad \forall (\mathbf{u}, \mathbf{v}) \in \mathcal{Z} \quad (15)$$

$$\mathbf{y}^v \in \mathcal{R}(\mathbf{x}) \quad \forall (\mathbf{u}, \mathbf{v}) \in \mathcal{Z} \quad (16)$$

$$\mathbf{x} \in \mathcal{X}' \quad (17)$$

The model (15)–(17) has an exponential number of variables and constraints. One can solve or approximate it by using a row and column generation technique (see, e.g., Zeng and Zhao 2013). The idea is to solve (15)–(17) for some subset $\mathcal{Z}' \subseteq \mathcal{Z}$ obtaining a solution $\mathbf{x}' \in \mathcal{X}'$, together with a lower bound on the optimal objective value. The upper bound and the cut indexed by $(\mathbf{u}', \mathbf{v}') \in \mathcal{Z}$, which can be added to $\mathcal{Z}'$, can be computed by solving the formulation (8)–(10) for $\mathbf{x}'$. Adding the cuts iteratively we can compute an exact or approximate solution to TSTR $\mathcal{P}$. The efficiency of this method can depend on the structure of $\mathcal{X}$ and should be verified experimentally for each particular case.

4 The shortest path problem

In this section we deal with the case of TSTR $\mathcal{P}$, in which $\mathcal{P}$ is the SHORTEST PATH problem. Let $G = (V, A)$ be a given network with two distinguished nodes $s \in V$ and $t \in V$. We will discuss two variants of the problem. In the first one, $\mathcal{X}_{\mathcal{P}}$ contains the characteristic vectors of all simple $s - t$ paths in G . In the second case, $\overline{\mathcal{X}}_{\mathcal{P}}$ contains all subsets of the arcs in which s and t are connected. Observe that $\mathcal{X}_{\mathcal{P}} \subseteq \overline{\mathcal{X}}_{\mathcal{P}}$ and in the deterministic case the problems

with both sets are equivalent. To see that the situation is different in the two-stage model, consider the sample instance shown in Fig. 2.

In the problem with set $\mathcal{X}_{\mathcal{P}}$ the maximum regret of each first-stage solution equals M . On the other hand, for the set $\bar{\mathcal{X}}_{\mathcal{P}}$ we can choose $\mathbf{x} = (1, 1, 0, 0)$, i.e. we can select the arcs $(s, 1)$ and $(s, 2)$ in the first stage. Then, depending on the second-stage scenario we can complete this solution by choosing $(1, t)$ or $(2, t)$. The maximum regret of $\mathbf{x}$ is then 0. The example demonstrates that it can be profitable to select more arcs in the first stage, even if some of them are not ultimately used. The next theorem describes the computational complexity of TSTR SHORTEST PATH.

Theorem 1 *The TSTR SHORTEST PATH problem with both $\mathcal{X}_{\mathcal{P}}$ and $\bar{\mathcal{X}}_{\mathcal{P}}$ is NP-hard.*

Proof Consider the NP-hard PARTITION problem defined as follows (Garey and Johnson 1979). We are given a collection $a_1, \dots, a_n$ of positive integers, such that $\sum_{i \in [n]} a_i = 2b$. We ask if there is a subset $I \subseteq [n]$ such that $\sum_{i \in I} a_i = b$. Given an instance of PARTITION we build the graph shown in Fig. 3. For each arc we specify the first-stage cost and the second-stage cost interval, where $M > 2nb + 2b$ is a sufficiently large constant (see Fig. 3). We show that the answer to PARTITION is yes if and only if there is a first-stage solution $\mathbf{x}$ with the maximum regret at most $\frac{3}{2}b$. We first focus on the set $\mathcal{X}_{\mathcal{P}}$. To this end, we prove the following three claims:

1. *There is an optimal first-stage solution with the maximum regret at most $2b$.* Let $\mathbf{x}$ be the first-stage solution in which all arcs $q_1, \dots, q_n$ are selected. Under any scenario, the optimal recourse action for $\mathbf{x}$ selects the arcs $q'_1, \dots, q'_n$. In the worst-case scenario $\underline{c}$ the

Fig. 2 An instance of TSTR SHORTEST PATH, where M is a sufficiently large number

Fig. 3 The instance of TSTR SHORTEST PATH from the proof of Theorem 1

- second-stage costs of all arcs are set to their lower bounds. We get $\text{Inc}(\mathbf{x}, \underline{\mathbf{c}}) = 2nb + 2b$, $\text{Opt}(\underline{\mathbf{c}}) = 2nb$ and $Z(\mathbf{x}) = 2b$.
2. The first-stage solution $\mathbf{x}$, in which no arc is chosen in the first stage or at least one of the arcs among $r, p'_i, q'_i, i \in [n]$, is chosen in the first stage is not optimal. Define scenario $\mathbf{c}_1$ in which the second-stage cost of arc r is M and the second-stage costs of the arcs $p'_1, \dots, p'_n$ are 0. Observe that $\text{Opt}(\mathbf{c}_1) = 2nb$. It is easy to see that $\text{Inc}(\mathbf{x}, \mathbf{c}_1) \geq M$. Hence $Z(\mathbf{x}) \geq M - 2nb > 2b$ and $\mathbf{x}$ is not optimal, according to point 1.
 3. Any optimal first-stage solution $\mathbf{x}$ selects exactly one of p_i or q_i for each $i \in [n]$. According to point 2, at least one of the arcs among $p_i, q_i, i \in [n]$, must be selected by $\mathbf{x}$. Assume there is $k \in [n]$ such that both p_k and q_k are not selected in $\mathbf{x}$. Consequently, we must choose p_k or q_k in the second stage which implies $\text{Inc}(\mathbf{x}, \underline{\mathbf{c}}) \geq M$. Since $\text{Opt}(\underline{\mathbf{c}}) = 2nb$, we get $Z(\mathbf{x}) \geq M - 2nb > 2b$ and $\mathbf{x}$ is not optimal, according to point 1.

Let $I \subseteq [n]$ be the set of indices of the arcs p_i selected in the first stage. Notice that $[n] \setminus I$ is the set of indices of the arcs q_i selected in the first stage. If arc p_i is chosen in the first stage, then p'_i must be chosen in the second state (the same is true for arcs q_i). In the worst-case scenario $\mathbf{c}$ we fix the cost of r to $2nb + b$ and the cost of p'_i to $\frac{3}{2}a_i$ if $i \in I$ and to 0, otherwise. We get $\text{Inc}(\mathbf{x}, \mathbf{c}) = 2nb + \sum_{i \in I} \frac{3}{2}a_i + \sum_{i \in [n] \setminus I} a_i$ and $\text{Opt}(\mathbf{c}) = 2nb + \min\{b, \sum_{i \in I} a_i\}$. The maximum regret of the formed path $\mathbf{x}$ is then

$$\begin{aligned} Z(\mathbf{x}) &= \max \left\{ \sum_{i \in [n] \setminus I} a_i + \sum_{i \in I} \frac{3}{2}a_i - \sum_{i \in I} a_i, \sum_{i \in [n] \setminus I} a_i + \sum_{i \in I} \frac{3}{2}a_i - b \right\} \\ &= \max \left\{ 2b - \frac{1}{2} \sum_{i \in I} a_i, b + \frac{1}{2} \sum_{i \in I} a_i \right\}. \end{aligned}$$

We now can see that the maximum regret of the formed path is at most $\frac{3}{2}b$ if and only if $\frac{1}{2} \sum_{i \in I} a_i = \frac{1}{2}b$, i.e. the answer to PARTITION is yes.

Let us now turn to the case of $\bar{\mathcal{X}}_{\mathcal{P}}$. Because $\mathcal{X}_{\mathcal{P}} \subseteq \bar{\mathcal{X}}_{\mathcal{P}}$, the positive answer to the PARTITION problem implies that there is a first-stage solution $\mathbf{x} \in \bar{\mathcal{X}}_{\mathcal{P}}$ such that $Z(\mathbf{x}) \leq \frac{3}{2}b$. It remains to show the converse implication, i.e. if there is a first-stage solution $\mathbf{x} \in \bar{\mathcal{X}}_{\mathcal{P}}$ such that $Z(\mathbf{x}) \leq \frac{3}{2}b$, then the answer to PARTITION is yes. In the case of $\bar{\mathcal{X}}_{\mathcal{P}}$ any subset of arcs in the formed network G is allowed to be selected in the first-stage. Similarly to the previous case, selecting any arc among $r, p'_i, q'_i, i \in [n]$, in the first stage yields a solution $\mathbf{x}$ such that $Z(\mathbf{x}) > 2b$. Hence, it may only be profitable to choose more than n arcs among $p_i, q_i, i \in [n]$ in the first stage. Let $\mathbf{x}$ be any such a solution. Under scenario $\underline{\mathbf{c}}$, we get $\text{Inc}(\mathbf{x}, \underline{\mathbf{c}}) \geq 2nb + 2b$, while $\text{Opt}(\underline{\mathbf{c}}) = 2nb$. Hence $Z(\mathbf{x}) \geq 2b$, a contradiction with the assumption that $Z(\mathbf{x}) \leq \frac{3}{2}b$. $\square$

Theorem 2 Computing the maximum regret $Z(\mathbf{x})$ for a given $\mathbf{x} \in \mathcal{X}'$ is NP-hard for both $\mathcal{X}_{\mathcal{P}}$ and $\bar{\mathcal{X}}_{\mathcal{P}}$.

Proof Given again an instance of the PARTITION problem (see the proof of Theorem 1). We construct a network $G = (V, A)$ consisting of two disjoint paths from s to t , P_1 and P_2 , each with n arcs. The network G with the corresponding first-stage costs and the second-stage cost intervals is shown in Fig. 4. Set $\mathbf{x} = \mathbf{0}$, so no arc is allowed to be selected in the first stage. We will show that the answer PARTITION is yes if and only if $Z(\mathbf{x}) \geq b$.

Let $\mathbf{c}$ be a worst-scenario for $\mathbf{x}$. According to Corollary 1, we can assume that $\mathbf{c}$ is an extreme scenario. Let I be the set of indices of the arcs in P_1 whose second-stage costs are

Fig. 4 The instance of TSTR SHORTEST PATH from the proof of Theorem 2

set to the upper bounds. The value of $\text{Inc}(\mathbf{x}, \mathbf{c})$ is the minimum of $\sum_{i \in [n]} a_i$ (path P_2 is used as the best recourse action) and $\sum_{i \in I} 2a_i$ (path P_1 is used as the best recourse action). On the other hand $\text{Opt}(\mathbf{c}) = \sum_{i \in I} a_i$, because the optimal two-stage path is P_1 , where the arcs with indices in I are selected in the first-stage. Hence

$$Z(\mathbf{x}) = \min \left\{ \sum_{i \in [n]} a_i, \sum_{i \in I} 2a_i \right\} - \sum_{i \in I} a_i = \min \left\{ \sum_{i \in [n] \setminus I} a_i, \sum_{i \in [n]} a_i \right\}.$$

Therefore, $Z(\mathbf{x}) \geq b$ if and only if the answer to PARTITION problem is yes. $\square$

In the following we analyze the computational complexity of the incremental and two-stage variants of the problem.

Observation 1 The INC SHORTEST PATH problem with $\bar{\mathcal{X}}_{\mathcal{P}}$ can be solved in polynomial time.

Proof Given network $G = (V, A)$, a first-stage solution $\mathbf{x}$ and a second-stage cost scenario $\mathbf{c}$, consider a deterministic SHORTEST PATH problem in the same network G , in which the costs of the arcs selected in $\mathbf{x}$ are 0 and the costs of the remaining arcs are determined according to $\mathbf{c}$. We seek a shortest $s - t$ path in G . The optimal recourse action $\mathbf{y} \in \mathcal{R}(\mathbf{x})$ selects the arcs on the path computed, which are not selected by $\mathbf{x}$. Notice that $\mathbf{x} + \mathbf{y}$ needs not to describe a simple $s - t$ path in G . $\square$

Observation 2 The TST SHORTEST PATH problem with both $\bar{\mathcal{X}}_{\mathcal{P}}$ and $\mathcal{X}_{\mathcal{P}}$ can be solved in polynomial time.

Proof Given a network $G = (V, A)$ in which C_i is the first-stage cost and c_i is the second-stage cost of the arc $a_i \in A$, consider a deterministic SHORTEST PATH problem in the same network G , in which the cost of the arc a_i is $\hat{c}_i = \min\{C_i, c_i\}$. Let P be the shortest $s - t$ path in G with the arc costs $\hat{c}_i$. We form solution $(\mathbf{x}, \mathbf{y}) \in \mathcal{Z}$ as follows: for each $a_i \in P$, if $\hat{c}_i = C_i$, then $x_i = 1$; if $\hat{c}_i < C_i$, then $y_i = 1$; for each $a_i \notin P$, $x_i = y_i = 0$. One can easily verify that $(\mathbf{x}, \mathbf{y})$ is an optimal solution to TST SHORTEST PATH regardless of which set $\bar{\mathcal{X}}_{\mathcal{P}}$ or $\mathcal{X}_{\mathcal{P}}$ is used. $\square$

The next result shows a difference between the two problem variants using $\mathcal{X}_{\mathcal{P}}$ and $\bar{\mathcal{X}}_{\mathcal{P}}$.

Theorem 3 The INC SHORTEST PATH problem with $\mathcal{X}_{\mathcal{P}}$ is strongly NP-hard and not at all approximable if $P \neq \text{NP}$.

Proof Consider the following strongly NP-complete HAMILTONIAN PATH problem (Garey and Johnson 1979). We are given a directed graph $G = (V, A)$ with two distinguished nodes

Fig. 5 A sample reduction in the proof of Theorem 3. Arcs with zero costs are solid. Dashed forward arcs were chosen in the first stage

$v_1, v_n \in V$. We ask if it is possible to find a directed path from v_1 to v_n that visits each node in V exactly once. We build a graph $G' = (V', A')$ as follows. For each $v_i \in V$, the node set V' contains two nodes v_i and v'_i . The set of arcs A' contains the *forward arcs* (v_i, v'_i) for all $v_i \in V$, and *backward arcs* (v'_i, v_j) for all $(v_i, v_j) \in A$. Finally, there are *dummy arcs* (v'_i, v_{i+1}) for all $i \in [n - 1]$. The first stage costs of all arcs in A' are 0. Under the second-stage scenario $\mathbf{c}$, the costs of all forward and backward arcs are 0 and the costs of all dummy arcs are 1. We set $s = v_1$ and $t = v_n$. Figure 5 shows a sample reduction.

The presence of dummy arcs ensures that $\mathbf{x}$ is feasible, as we can complete $\mathbf{x}$ to a simple path by using all dummy arcs. Now it is easy to see that there is an optimal recourse action $\mathbf{y} \in \mathcal{R}(\mathbf{x})$, which selects only the backward arcs if and only if there is a Hamiltonian path in G . In other words, there is $\mathbf{y} \in \mathcal{R}(\mathbf{x})$ such that $\text{Inc}(\mathbf{x}, \mathbf{c}) = 0$ if and only if G has a Hamiltonian path, which proves the theorem. $\square$

5 The selection problem

In this section we discuss the TSTR SELECTION problem, in which $\mathcal{X}_S = \{\mathbf{x} \in \{0, 1\}^n : x_1 + \dots + x_n = p\}$ for some fixed $p \in [n]$. We can interpret $\mathbf{x} \in \mathcal{X}_S$ as a characteristic vector of a selection of exactly p items out of n available. The set of feasible first-stage solutions is then $\mathcal{X}'_S = \{\mathbf{x} \in \{0, 1\}^n : x_1 + \dots + x_n \leq p\}$. The deterministic SELECTION problem can be solved in $O(n)$ time. We first find in $O(n)$ time the p th smallest item cost c (see, e.g., Cormen et al. 1990) and select p items with the costs at most c . It is easy to see that the corresponding TST SELECTION and INC SELECTION problems are solvable in $O(n)$ time as well. Various robust versions of SELECTION have been discussed in the literature. In particular, the single-stage minmax and minmax regret models were investigated in Averbakh (2001), Conde (2004), Doerr (2013) and Kasperski et al. (2013) and the robust two-stage models, with the minmax criterion, were discussed in Chassein et al. (2018) and Kasperski and Zieliński (2017).

5.1 Computing the maximum regret

In this section we show that the value of $Z(\mathbf{x})$ for a given $\mathbf{x} \in \mathcal{X}'_S$ can be computed in polynomial time. Consider the subproblem $\min_{\mathbf{y} \in \mathcal{R}(\mathbf{x})} \mathbf{c}_v^T \mathbf{y}$ from Proposition 1, which can be represented as the following linear programming problem:

$$\min \mathbf{c}_v^T \mathbf{y} \quad (18)$$

$$\text{s.t. } \sum_{i \in [n]} y_i \geq p - \sum_{i \in [n]} x_i \quad (19)$$

$$y_i \leq 1 - x_i \quad \forall i \in [n] \quad (20)$$

$$y_i \geq 0 \quad \forall i \in [n] \quad (21)$$

Observe that we have relaxed the constraints $y_i \in \{0, 1\}$, $i \in [n]$, without changing the optimal objective function value, because the problem (18)–(21) has an integral optimal solution, which is due to total unimodularity of the constraint matrix (19) and (20). Using the definition of $\mathbf{c}_v$ (see Proposition 1), we can write the dual to (18)–(21), which is another linear programming problem of the form:

$$\begin{aligned} \max & (p - \sum_{i \in [n]} x_i) \alpha + \sum_{i \in [n]} (x_i - 1) \beta_i \\ \text{s.t. } & \alpha - \beta_i \leq \underline{c}_i v_i + \bar{c}_i (1 - v_i) \quad \forall i \in [n] \\ & \alpha, \beta_i \geq 0 \quad \forall i \in [n] \end{aligned}$$

Using Proposition 1 and equality (3), we can build the following compact mixed integer programming formulation for computing the maximum regret of a given first-stage solution $\mathbf{x} \in \mathcal{X}'_S$:

$$Z(\mathbf{x}) = \max \mathbf{C}^T \mathbf{x} - \mathbf{C}^T \mathbf{u} - \underline{c}^T \mathbf{v} + (p - \sum_{i \in [n]} x_i) \alpha + \sum_{i \in [n]} (x_i - 1) \beta_i \quad (22)$$

$$\text{s.t. } \alpha - \beta_i \leq \underline{c}_i v_i + \bar{c}_i (1 - v_i) \quad \forall i \in [n] \quad (23)$$

$$\sum_{i \in [n]} (u_i + v_i) = p \quad (24)$$

$$u_i + v_i \leq 1 \quad \forall i \in [n] \quad (25)$$

$$u_i, v_i \in \{0, 1\} \quad \forall i \in [n] \quad (26)$$

$$\alpha, \beta_i \geq 0 \quad \forall i \in [n] \quad (27)$$

The constraints (24)–(26) represent a feasible pair $(\mathbf{u}, \mathbf{v}) \in \mathcal{Z}$. Define $X = \{i \in [n] : x_i = 1\}$ and let $\bar{X} = \{i \in [n] : x_i = 0\}$ be the complement of X . Observe that in an optimal solution to (22)–(27) we can fix $\beta_i = [\alpha - \underline{c}_i v_i - \bar{c}_i (1 - v_i)]_+$ for each $i \in [n]$ (we use the notation $[t]_+ = \max\{0, t\}$). As $v_i \in \{0, 1\}$, we can set:

$$\sum_{i \in [n]} (x_i - 1) \beta_i = - \sum_{i \in \bar{X}} ([\alpha - \underline{c}_i]_+ - [\alpha - \bar{c}_i]_+) v_i - \sum_{i \in \bar{X}} [\alpha - \bar{c}_i]_+. \quad (28)$$

Denote $c_i^*(\alpha) = [\alpha - \underline{c}_i]_+ - [\alpha - \bar{c}_i]_+$ and, using (28), rewrite (22)–(27) as

$$Z(\mathbf{x}) = \max \mathbf{C}^T \mathbf{x} - \mathbf{C}^T \mathbf{u} - \underline{c}^T \mathbf{v} + (p - |X|) \alpha - \sum_{i \in \bar{X}} c_i^*(\alpha) v_i - \sum_{i \in \bar{X}} [\alpha - \bar{c}_i]_+ \quad (29)$$

$$\text{s.t. } \sum_{i \in [n]} (u_i + v_i) = p \quad (30)$$

$$u_i + v_i \leq 1 \quad \forall i \in [n] \quad (31)$$

$$u_i, v_i \in \{0, 1\} \quad \forall i \in [n] \quad (32)$$

$$\alpha \geq 0 \quad (33)$$

Define $\mathcal{C} = \{\underline{c}_1, \dots, \underline{c}_n\} \cup \{\bar{c}_1, \dots, \bar{c}_n\}$.

Lemma 1 *There is an optimal solution to (29)–(33) in which $\alpha \in \mathcal{C}$.*

Proof Fix any $(\mathbf{u}, \mathbf{v}) \in \mathcal{Z}$ to (29)–(33). The optimal value of $\alpha \geq 0$ can be then found by solving the following problem:

$$\max_{\alpha \geq 0} \left((p - |X|)\alpha - \sum_{i \in \bar{X}} [\alpha - \hat{c}_i]_+ \right), \quad (34)$$

where $\hat{c}_i = \underline{c}_i$ if $v_i = 1$ and $\hat{c}_i = \bar{c}_i$, otherwise. Since $|\bar{X}| \geq p - |X|$, the problem (34) attains a maximum at some $\alpha \geq 0$. As the objective function of this problem is piecewise linear, the optimal value of α is at some $\hat{c}_k$, $k \in \bar{X}$. Since $\hat{c}_k \in \mathcal{C}$, the lemma follows. $\square$

Theorem 4 *The value of $Z(\mathbf{x})$ for a given $\mathbf{x} \in \mathcal{X}'_S$ can be computed in $O(n^2)$ time.*

Proof Fix $\alpha \in \mathcal{C}$ in (29)–(33). The remaining optimization problem is then

$$\min \mathbf{C}^T \mathbf{u} + \hat{\mathbf{c}}^T \mathbf{v} \quad (35)$$

$$\text{s.t. } \sum_{i \in [n]} (u_i + v_i) = p \quad (36)$$

$$u_i + v_i \leq 1 \quad \forall i \in [n] \quad (37)$$

$$u_i, v_i \in \{0, 1\} \quad \forall i \in [n], \quad (38)$$

where $\hat{c}_i = \underline{c}_i$ if $i \notin \bar{X}$ and $\hat{c}_i = \underline{c}_i + [\alpha - \underline{c}_i]_+ - [\alpha - \bar{c}_i]_+$, otherwise. Observe that (35)–(38) is a TST SELECTION problem with the first-stage costs $\mathbf{C}$ and the second-stage costs $\hat{\mathbf{c}}$, which can be solved in $O(n)$ time. By Lemma 1 it is enough to try at most $2n$ values of α in $\mathcal{C}$ to find an optimal solution to (29)–(33). Therefore, one can solve this problem and thus compute $Z(\mathbf{x})$ in $O(n^2)$ time. $\square$

5.2 Compact MIP formulations

Fix $\alpha \in \mathcal{C}$ in (29)–(33). One can easily check that the constraint matrix (30) and (31) is totally unimodular. Hence we can relax the constraints $u_i, v_i \in \{0, 1\}$ with $u_i, v_i \geq 0$ and write the following dual to the relaxed (29)–(33):

$$\begin{aligned} \min \quad & \mathbf{C}^T \mathbf{x} + (p - |X|)\alpha - \sum_{i \in [n]} [\alpha - \bar{c}_i]_+ (1 - x_i) + p\pi(\alpha) + \sum_{i \in [n]} \rho_i(\alpha) \\ \text{s.t.} \quad & \pi(\alpha) + \rho_i(\alpha) \geq -C_i \quad \forall i \in [n] \\ & \pi(\alpha) + \rho_i(\alpha) \geq -\underline{c}_i - c_i^*(\alpha)(1 - x_i) \quad \forall i \in [n] \end{aligned}$$

$$\rho_i(\alpha) \geq 0 \quad \forall i \in [n]$$

which can be rewritten as:

$$\begin{aligned} \min \quad & \mathbf{C}^T \mathbf{x} + (p - \sum_{i \in [n]} x_i) \alpha + \sum_{i \in [n]} [\alpha - \bar{c}_i]_+ (x_i - 1) - p\pi(\alpha) + \sum_{i \in [n]} \rho_i(\alpha) \\ \text{s.t.} \quad & \pi(\alpha) - \rho_i(\alpha) \leq C_i \quad \forall i \in [n] \\ & \pi(\alpha) - \rho_i(\alpha) \leq \underline{c}_i + c_i^*(\alpha)(1 - x_i) \quad \forall i \in [n] \\ & \rho_i(\alpha) \geq 0 \quad \forall i \in [n] \end{aligned}$$

We can thus construct the following MIP formulation for TStR SELECTION:

$$\min \quad \mathbf{C}^T \mathbf{x} + z \quad (39)$$

$$\text{s.t. } z \geq (p - \sum_{i \in [n]} x_i) \alpha + \sum_{i \in [n]} [\alpha - \bar{c}_i]_+ (x_i - 1) - p\pi(\alpha) + \sum_{i \in [n]} \rho_i(\alpha) \quad \alpha \in \mathcal{C} \quad (40)$$

$$\pi(\alpha) - \rho_i(\alpha) \leq C_i \quad \forall i \in [n], \alpha \in \mathcal{C} \quad (41)$$

$$\pi(\alpha) - \rho_i(\alpha) \leq \underline{c}_i + c_i^*(\alpha)(1 - x_i) \quad \forall i \in [n], \alpha \in \mathcal{C} \quad (42)$$

$$\sum_{i \in [n]} x_i \leq p \quad (43)$$

$$\rho_i(\alpha) \geq 0 \quad \forall i \in [n], \alpha \in \mathcal{C} \quad (44)$$

$$x_i \in \{0, 1\} \quad i \in [n] \quad (45)$$

Here, variables $x_i \in \{0, 1\}$ have been added to represent the choice of first-stage solution, along with constraint (44) to model that $\mathbf{x} \in \mathcal{X}'_S$. Taking the minimum over the new variable z gives the maximum over $\alpha \in \mathcal{C}$ by an epigraph reformulation.

In the following we will show how to decompose (40)–(45) into a family of problems of smaller size. The key idea will be to show that it is enough to try at most $O(n^2)$ cases for the vector of variables $\boldsymbol{\pi} = (\pi(\alpha))_{\alpha \in \mathcal{C}}$.

Lemma 2 *There exist $\hat{c}_k \in \mathcal{A} = \{C_1, \dots, C_n\} \cup \{\underline{c}_1, \dots, \underline{c}_n\}$ and $\hat{c}_l \in \mathcal{B} = \{C_1, \dots, C_n\} \cup \{\underline{c}_1, \dots, \underline{c}_n\} \cup \{\bar{c}_1, \dots, \bar{c}_n\}$, $\hat{c}_k \leq \hat{c}_l$ such that*

$$\pi(\alpha) = \max\{\hat{c}_k, \min\{\alpha, \hat{c}_l\}\} = \begin{cases} \hat{c}_k & \text{if } \alpha \leq \hat{c}_k \\ \alpha & \text{if } \hat{c}_k < \alpha < \hat{c}_l, \alpha \in \mathcal{C} \\ \hat{c}_l & \text{if } \hat{c}_l \leq \alpha \end{cases}$$

are optimal to (40)–(45).

Proof Let us fix $\mathbf{x}$ and α in (40)–(45), and define $r_i(\alpha, x_i) = \min\{C_i, \underline{c}_i + c_i^*(\alpha)(1 - x_i)\} = \min\{C_i, \underline{c}_i + ([\alpha - \underline{c}_i]_+ - [\alpha - \bar{c}_i]_+)(1 - x_i)\}$. For each $\alpha \in \mathcal{C}$, the values of $\pi(\alpha)$ and $\rho_i(\alpha)$ can then be found by solving the following linear programming problem:

$$\max \quad p\pi(\alpha) - \sum_{i \in [n]} \rho_i(\alpha)$$

$$\begin{aligned} \text{s.t. } \pi(\alpha) - \rho_i(\alpha) &\leq r_i(\alpha, x_i) & \forall i \in [n] \\ \rho_i(\alpha) &\geq 0 & \forall i \in [n] \end{aligned}$$

By using linear programming duality, one can check that in an optimal solution to this problem, we can set $\pi(\alpha)$ to the p th smallest value among $r_i(\alpha, x_i)$, $i \in [n]$, meaning that $\pi(\alpha) \in \{r_i(\alpha, x_i) : i \in [n]\}$. We now consider all possible shapes of $r_i(\alpha, x_i)$.

1. $r_i(\alpha, 1) = \min\{C_i, \underline{c}_i\}$,
2. If $C_i \leq \underline{c}_i \leq \bar{c}_i$, then $r_i(\alpha, 0) = C_i$.
3. If $\underline{c}_i \leq C_i \leq \bar{c}_i$, then

$$r_i(\alpha, 0) = \begin{cases} \underline{c}_i & \text{if } \alpha \leq \underline{c}_i \\ \alpha & \text{if } \underline{c}_i < \alpha < C_i \\ C_i & \text{if } C_i \leq \alpha \end{cases}$$

4. If $\underline{c}_i \leq \bar{c}_i \leq C_i$, then

$$r_i(\alpha, 0) = \begin{cases} \underline{c}_i & \text{if } \alpha \leq \underline{c}_i \\ \alpha & \text{if } \underline{c}_i < \alpha < \bar{c}_i \\ \bar{c}_i & \text{if } \bar{c}_i \leq \alpha \end{cases}$$

The three cases for $r_i(\alpha, 0)$ are visualized in Fig. 6. All possible shapes have in common that they have a constant value in $\{C_i, \underline{c}_i\}$ up to the diagonal. They then follow the diagonal to leave at another constant value in $\{C_i, \underline{c}_i, \bar{c}_i\}$. This means that the function representing the p th smallest value over all $r_i(\alpha, x_i)$ is also of this shape.

Hence, we can enumerate all possible combinations of values $(\hat{c}_k, \hat{c}_l)$ with $\hat{c}_k \leq \hat{c}_l$, $\hat{c}_k \in \{C_1, \dots, C_n\} \cup \{\underline{c}_1, \dots, \underline{c}_n\} = \mathcal{A}$ and $\hat{c}_l \in \{C_1, \dots, C_n\} \cup \{\underline{c}_1, \dots, \underline{c}_n\} \cup \{\bar{c}_1, \dots, \bar{c}_n\} = \mathcal{B}$ and to define candidate functions $\pi(\alpha)$ as being equal to $\hat{c}_k$ if $\alpha \leq \hat{c}_k$, equal to α if $\alpha \in [\hat{c}_k, \hat{c}_l]$, and equal to $\hat{c}_l$ if $\alpha \geq \hat{c}_l$, as claimed. Amongst these candidate functions, an optimal choice of $\pi(\alpha)$ is then contained. $\square$

Using Lemma 2 we can enumerate all vectors $\boldsymbol{\pi} = (\pi(\alpha))_{\alpha \in \mathcal{C}}$ and denote the set of these vectors by Π . Notice that $|\Pi|$ is $O(n^2)$. For a fixed $\boldsymbol{\pi} = (\pi(\alpha))_{\alpha \in \mathcal{C}} \in \Pi$, we can rewrite the problem (40)–(45) as follows:

$$\begin{aligned} P(\boldsymbol{\pi}) &= \min z \\ \text{s.t. } z &\geq v^{\boldsymbol{\pi}}(\alpha) + \sum_{i \in [n]} \omega_i^{\boldsymbol{\pi}}(\alpha) x_i & \forall \alpha \in \mathcal{C} \end{aligned}$$

Fig. 6 Shape of $r_i(\alpha, 0)$ with $\underline{c}_i = 3$, $\bar{c}_i = 7$

$$\sum_{i \in [n]} x_i \leq p$$

$$x_i \in \{0, 1\} \quad \forall i \in [n]$$

where

$$\nu^\pi(\alpha) = p\alpha - \sum_{i \in [n]} [\alpha - \bar{c}_i]_+ - p\pi(\alpha) + \sum_{i \in [n]} \underline{\rho}_i^\pi(\alpha)$$

$$\omega_i^\pi(\alpha) = C_i - \alpha + [\alpha - \bar{c}_i]_+ + \bar{\rho}_i^\pi(\alpha) - \underline{\rho}_i^\pi(\alpha)$$

and

$$\underline{\rho}_i^\pi(\alpha) = \max\{0, \pi(\alpha) - C_i, \pi(\alpha) - \underline{c}_i - [\alpha - \underline{c}_i]_+ + [\alpha - \bar{c}_i]_+\}$$

$$\bar{\rho}_i^\pi(\alpha) = \max\{0, \pi(\alpha) - C_i, \pi(\alpha) - \underline{c}_i\}$$

are constant values. We therefore find that

$$\min_{\mathbf{x} \in \mathcal{X}'} Z(\mathbf{x}) = \min_{\boldsymbol{\pi} \in \Pi} P(\boldsymbol{\pi}). \quad (46)$$

According to Lemma 2 we enumerate $O(n^2)$ many candidate vectors $\boldsymbol{\pi} \in \Pi$. We then solve the resulting problem with n binary variables and $O(n)$ constraints, which is substantially smaller than the MIP formulation (40)–(45). In the next section we will propose a heuristic greedy algorithm, which is based on the decomposition (46).

The computational complexity of the general TSTR SELECTION problem remains open. In the following we will identify some of its special cases which can be solved in polynomial time.

Proposition 2 *If $p = n$, then TSTR SELECTION can be solved in polynomial time.*

Proof For each $\mathbf{x} \in \mathcal{X}'$ and $\mathbf{y} \in \mathcal{R}(\mathbf{x})$, we get $\mathbf{x} + \mathbf{y} = n$. Hence for each $i \in [n]$, if $x_i = 1$, then the contribution of i to the maximum regret is $C_i - \min\{C_i, \underline{c}_i\}$ and if $x_i = 0$, then the contribution is $\bar{c}_i - \min\{C_i, \bar{c}_i\}$. Hence we set $x_i = 1$ if $C_i - \min\{C_i, \underline{c}_i\} \leq \bar{c}_i - \min\{C_i, \bar{c}_i\}$ for every $i \in [n]$. $\square$

Proposition 3 *The TSTR SELECTION problem can be solved in polynomial time if two out of the following three conditions hold:*

- (1) *The set $\{C_i : i \in [n]\}$ is of constant size.*
- (2) *The set $\{\underline{c}_i : i \in [n]\}$ is of constant size.*
- (3) *The set $\{\bar{c}_i : i \in [n]\}$ is of constant size.*

Proof We first consider the case in which the conditions (1) and (2) hold. Let the sets T_j , $j \in [\ell]$, consist of all $i \in [n]$ with the same values C_i and $\underline{c}_i$. Due to (1) and (2), ℓ is constant as well. Let $(l_1, \dots, l_\ell)$ be a vector of nonnegative integers such that $l_1 + \dots + l_\ell = p$. This vector defines a decomposition into subproblems where we pick l_j many items out of each set T_j . These items $i \in T_j$ only differ with respect to their upper bounds $\bar{c}_i$. An optimal solution is hence to pick those l_j many items i in the first stage that have the highest second-stage costs $\bar{c}_i$. The number of the vectors $\mathbf{l}$ enumerated is $O(n^\ell)$ so it remains polynomial. The cases that assumptions (1) and (3) as well as assumptions (2) and (3) hold can be treated in the same way. $\square$

5.3 Greedy algorithm

In this section we propose a heuristic algorithm for computing a solution to TSTR SELECTION, which can be applied to larger instances. The first idea consists in applying the mid-point scenario heuristic, i.e. to solve TST SELECTION under scenario $\mathbf{c}^m$ such that $c_i^m = (\underline{c}_i + \bar{c}_i)/2$ for each $i \in [n]$. Unfortunately, the approximation ratio of this algorithm is unbounded, which can be easily demonstrated by using an instance analogous to that in Fig. 1b. Observe that the TSTR SHORTEST PATH instance in this figure can be seen as an instance of TSTR SELECTION with $n = 2$ and $p = 1$. We will now propose a more complex heuristic for the problem, which is based on Eq. (46). Given $X \subseteq [n]$, let us define

$$F^\pi(X) = \max_{\alpha \in \mathcal{C}} \left(v^\pi(\alpha) + \sum_{i \in X} \omega_i^\pi(\alpha) \right).$$

Using (46), we get

$$\min_{\mathbf{x} \in \mathcal{X}'} Z(\mathbf{x}) = \min_{\pi \in \Pi} \min_{\{X: X \subseteq [n], |X| \leq p\}} F^\pi(X).$$

Theorem 5 *Function F^π is supermodular, i.e. for each $X \subseteq Y \subseteq [n]$ and $j \in [n] \setminus Y$ the inequality $F^\pi(Y \cup \{j\}) - F^\pi(Y) \geq F^\pi(X \cup \{j\}) - F^\pi(X)$ holds.*

Proof See the “Appendix”. $\square$

The greedy algorithm considers all possible $\pi \in \Pi$. For each fixed π we start with $X = \emptyset$ and greedily add the elements $i \in [n] \setminus X$ to X as long as an improvement is possible, i.e. if there is $i \in [n] \setminus X$ such that $F^\pi(X \cup \{i\}) < F^\pi(X)$. For a fixed choice of π , the greedy algorithm thus evaluates the objective F^π $O(np)$ many times. In total, there are therefore $O(n^3p)$ calls to F^π .

Theorem 5 allows us to reduce the search space of the algorithm. Namely, if adding j to X does not decrease the value of F^π at some step, then adding j to the current solution in the subsequent steps cannot improve the current solution either. Hence j can be removed from further consideration.

In order to illustrate the algorithm consider an instance of the problem shown in Table 1.

We get $\mathcal{A} = \{1, 2, 4, 6, 9, 12\}$, $\mathcal{B} = \{1, 2, 4, 6, 9, 12, 13\}$ and $\mathcal{C} = \{1, 2, 4, 6, 9, 12, 13\}$. For $\hat{c}_k = 2$ and $\hat{c}_l = 6$, we get $\pi(1) = 2$, $\pi(2) = 2$, $\pi(4) = 4$, $\pi(6) = 6$, $\pi(9) = 6$, $\pi(12) = 6$, and $\pi(13) = 6$. For this vector π we compute $P(\pi)$ by solving the following problem:

$$\begin{aligned} P(\pi) = \min \quad & z \\ \text{s.t.} \quad & z \geq -2 + 5x_1 + 0x_2 + 3x_3 + 11x_4 \\ & z \geq 1 + 4x_1 - 1x_2 + 2x_3 + 10x_4 \\ & z \geq 3 + 2x_1 - 3x_2 + 2x_3 + 10x_4 \\ & z \geq 5 + 0x_1 - 3x_2 + 0x_3 + 10x_4 \\ & z \geq 8 - 3x_1 - 3x_2 - 3x_3 + 10x_4 \\ & z \geq 11 - 6x_1 - 3x_2 - 6x_3 + 10x_4 \\ & z \geq 11 - 7x_1 - 3x_2 - 6x_3 + 10x_4 \\ & x_1 + x_2 + x_3 + x_4 \leq 3 \\ & x_1, x_2, x_3, x_4 \in \{0, 1\} \end{aligned}$$

Algorithm 1: A greedy algorithm for RTSR SELECTION.

```

1  $val^* \leftarrow \infty, X^* \leftarrow \emptyset;$ 
2 foreach  $\{(\hat{c}_k, \hat{c}_l) \in \mathcal{A} \times \mathcal{B} : \hat{c}_k \leq \hat{c}_l\}$  do
3   foreach  $\alpha \in \mathcal{C}$  do  $\pi(\alpha) \leftarrow \max\{\hat{c}_k, \min\{\alpha, \hat{c}_l\}\}$   $X \leftarrow \emptyset, bestval \leftarrow F^\pi(X), bestX \leftarrow X,$ 
    $improve \leftarrow true;$ 
4   while  $improve = true$  and  $|X| < p$  do
5      $improve \leftarrow false;$ 
6     foreach  $i \in [n] \setminus X$  do
7        $Y \leftarrow X \cup \{i\};$ 
8       if  $F^\pi(Y) \leq bestval$  then
9          $bestX \leftarrow Y;$ 
10         $bestval \leftarrow F^\pi(Y);$ 
11         $improve \leftarrow true;$ 
12     $X \leftarrow bestX;$ 
13  if  $bestval \leq val^*$  then
14     $val^* \leftarrow bestval;$ 
15     $X^* \leftarrow bestX;$ 
16 return  $X^*$ 

```

Table 1 Example problem for the greedy algorithm with $n = 4$ and $p = 3$

i	C_i	c_i	$\bar{c}_i$
1	6	9	13
2	1	1	4
3	4	2	12
4	12	2	6

An optimal solution to this problem is $\mathbf{x} = (0, 1, 1, 0)$ with $P(\pi) = 2$. In fact, this is an optimal first-stage solution to the sample instance with $Z(\mathbf{x}) = 2$. Figure 7 shows the search space of the greedy algorithm for the fixed π . If 1 is chosen in the first step, then the best achievable regret value is 4 by using the first-stage solution $X = \{1, 2\}$. However, the optimal regret value is 2 by using the first-stage solution $X = \{2, 3\}$. So, the example demonstrates that the approximation ratio of the greedy algorithm is at least 2. We conjecture that the algorithm is indeed a 2-approximation, so the example presented is a worst one. The search space can be reduced by applying Theorem 5. For example, 4 need only be considered in the first step, because $F^\pi(\emptyset) < F^\pi(\{4\})$. So adding 4 in the next steps cannot decrease the value of the current solution and 4 can be removed from further considerations. Also we need not to consider adding 3 to $\{1, 2\}$, because by Theorem 5, $F^\pi(\{1, 2\} \cup \{3\}) - F^\pi(\{1, 2\}) \geq F^\pi(\{1\} \cup \{3\}) - F^\pi(\{1\}) > 0$.

The example suggests an improvement of the greedy algorithm. Observe that we can achieve the optimal solution by adding 3 instead of 1 to $X = \emptyset$. So it may be advantageous to start from all possible subsets of $[n]$ with up to L items. For a small constant L , the algorithm remains polynomial.

Fig. 7 Greedy search space for the sample problem

6 Experimental results

In this section we show results of some computational tests for the TSTR SELECTION problem discussed in Sect. 5. We will compare various MIP formulations for this problem and the performance of the greedy heuristic described in Sect. 5.3.

6.1 Setup

We conduct computational experiments to answer the following questions:

1. Among the compact formulation (40)–(45), the decomposition formulation (46), and the general iterative row-and-column generation algorithm from Sect. 3, which approach is most efficient for the exact solution of two-stage min-max regret selection problems? How does this comparison depend on the instance size?
2. What is the performance of the greedy algorithm?

To answer these questions, we perform three experiments on instances generated in the following way. For each item i , costs C_i , c_i , and $\bar{c}_i$ are uniformly randomly taken from $\{1, \dots, r\}$, where we use $r = 20$ and $r = 100$. If $\bar{c}_i < c_i$, the two values are swapped. For experiments 1 and 2, we use instance sizes $n \in \{20, 40, \dots, 200\}$ and always set $p = n/2$. For experiment 3, we use $n \in \{300, 400, \dots, 1000\}$. We generate 100 instances per choice of n and r , which results in a total of 3600 instances.

For the iterative method, we solve the adversarial problem using the combinatorial algorithm used in the proof of Theorem 4. In preliminary experiments, this was more efficient than employing a MIP approach. For the greedy algorithm, we make use of Theorem 5 to search remaining items more efficiently.

The experiments were performed on a virtual Ubuntu server with ten Xeon CPU E7-2850 processors at 2.00 GHz speed and 23.5 GB RAM. We used CPLEX 12.8 to solve mixed-integer programming formulations, restricted to a single thread. Code is written in C++.

6.2 Experiment 1: exact solution algorithms

We compare the computation times for selection problems using the compact formulation [“Compact”—(40)–(45)], the decomposition formulation [“Decomp”—(46)] and using the general iterative row-and-column generation method (‘‘Iterative’’—Sect. 3). We start the row-

and-column generation with three cuts, corresponding to three adversary solutions. These are the respective optimal solutions in the following three scenarios: (i) when $\mathbf{c} = \underline{\mathbf{c}}$ and $\mathbf{y}^1 = \mathbf{0}$; (ii) when $\mathbf{y}^2 = \mathbf{0}$; and (iii) when all item costs are the minimum of C_i and $\underline{c}_i$, and we choose the p items with smallest costs. A time limit of 600 s was used for the iterative method.

In Table 2 we show median, average and maximum computation times (in seconds) for these three methods, as well as the standard deviation.

Overall, we find that Compact is several orders of magnitude faster than Iterative. While instances with $r = 20$ are consistently solved in less than a second using Compact, more than half of instances with size $n = 200$ reach the time limit of 600 s for Iterative. Furthermore, the computation times for Iterative have a high fluctuation, as can be seen in column “Stdev” and in the large difference between average and median times, while the computation times for Compact are stable. For both methods, the instances with $r = 100$ tend to be harder than the instances with $r = 20$. Note that larger values for r increase the size of the set $\mathcal{C}$ used in the compact formulation. We also observe that Compact is faster than Decomp on all instance types.

In Table 3 we show the number of cases (out of 100) that method Iterative was able to find an optimal solution for each instance size. Additionally, column “#It” shows how many iterations were performed on average (note that this value can be truncated by the time limit). We also show two gap values to measure the distance to optimality. The value “LB Gap” is the optimum objective value divided by the best lower bound found by Iterative, averaged only over the instances that were suboptimal, and given in percent. The value “UB Gap” is analogous with the best upper bound divided by the optimum objective value. Our results indicate that both LB and UB gap are relevant, i.e. Iterative does not find an optimal solution, and also the lower bound does reach the optimal value.

6.3 Experiment 2: greedy algorithm

In our second experiment, we evaluate the performance of the greedy algorithm. In Table 4 we present both computation times (“Median”, “Average”, “Stdev”, “Max”) as well as the number of instances where an optimal solution was found (“Opt”, note that the greedy method does not prove optimality), the average UB gap on suboptimal solutions, and the maximum UB gap in percent.

We first note that computation times for $r = 100$ tend to be higher than for method Compact. On the one hand, the implementation of the greedy algorithm depends on the skill of the programmer (whereas Compact makes use of CPLEX). On the other hand, the main advantage of the greedy algorithm is the theoretical bound on the computation time, which remains polynomial in the input size, while this guarantee cannot be given for Compact. Note that the algorithm is started for every possible value of π , of which there are $O(\min\{r, n\}^2)$ many.

Considering the quality of solutions, we see that the number of cases where an optimal solution can be identified drops with the instance size. At the same time, the UB gap also becomes smaller with larger n , meaning that more instances are suboptimal, but the relative gap to the optimal objective value becomes smaller. Note that larger instances have a higher objective value, which can explain why wrong decision on very small instances can lead to larger relative differences. Over all 2000 instances, the largest gap we observed was 21.4%, which is well below the gap of 100% generated by the instance we constructed in Table 1. Overall, the results give numerical evidence that the greedy algorithm may indeed give an approximation guarantee.

Table 2 Computation times (in seconds) for exact approaches

r	p	n	Median			Average		
			Compact	Decomp	Iterative	Compact	Decomp	Iterative
20	10	20	0.01	0.27	0.01	0.01	0.26	0.03
20	20	40	0.02	0.43	0.04	0.02	0.42	0.19
20	30	60	0.04	0.53	0.14	0.04	0.52	2.38
20	40	80	0.04	0.60	0.57	0.05	0.59	8.74
20	50	100	0.06	0.69	1.46	0.07	0.69	12.55
20	60	120	0.07	0.76	7.50	0.09	0.76	90.67
20	70	140	0.10	0.85	17.77	0.11	0.85	134.61
20	80	160	0.10	0.93	71.49	0.13	0.93	218.26
20	90	180	0.16	0.99	140.32	0.19	1.00	280.36
20	100	200	0.14	1.07	600.00	0.21	1.08	358.14
100	10	20	0.02	1.77	0.02	0.02	1.75	0.04
100	20	40	0.08	8.03	0.08	0.09	8.17	1.09
100	30	60	0.19	19.24	0.35	0.21	19.39	7.13
100	40	80	0.37	33.13	1.56	0.41	33.26	27.88
100	50	100	0.58	51.61	8.10	0.63	50.85	69.44
100	60	120	0.96	65.46	24.05	1.00	65.36	123.90
100	70	140	1.40	82.83	169.23	1.40	82.96	302.53
100	80	160	1.74	100.79	537.73	1.74	99.80	351.92
100	90	180	2.09	115.31	600.00	2.15	114.30	448.70
100	100	200	2.44	130.55	600.00	2.59	129.71	500.61
			Stdev			Max		
			Compact	Decomp	Iterative	Compact	Decomp	Iterative
20	10	20	0.00	0.06	0.07	0.03	0.38	0.66
20	20	40	0.01	0.05	0.43	0.05	0.52	2.69
20	30	60	0.02	0.05	6.57	0.13	0.63	35.13
20	40	80	0.02	0.05	59.51	0.12	0.72	595.30
20	50	100	0.02	0.06	33.93	0.18	0.80	278.45
20	60	120	0.04	0.07	185.55	0.25	0.99	600.00
20	70	140	0.04	0.07	209.10	0.28	1.00	600.00
20	80	160	0.08	0.07	254.06	0.46	1.09	600.00
20	90	180	0.15	0.08	266.21	0.97	1.17	600.00
20	100	200	0.15	0.08	270.73	0.92	1.22	600.00
100	10	20	0.01	0.32	0.09	0.05	2.36	0.84
100	20	40	0.03	1.47	7.00	0.18	12.06	69.06
100	30	60	0.06	3.02	45.59	0.41	26.53	447.27
100	40	80	0.18	4.95	96.61	1.77	53.86	600.00
100	50	100	0.17	6.28	146.69	1.06	64.24	600.00
100	60	120	0.25	8.03	190.91	1.99	80.91	600.00
100	70	140	0.32	8.16	272.99	2.72	101.05	600.00
100	80	160	0.43	9.51	265.87	3.38	127.81	600.00

Table 2 continued

			Stdev			Max		
			Compact	Decomp	Iterative	Compact	Decomp	Iterative
100	90	180	0.58	10.10	230.59	6.26	136.85	600.00
100	100	200	0.80	11.05	192.63	7.71	156.55	600.00

Table 3 Quality of solutions found by iterative method

r	p	n	#It	Opt	LB Gap	UB Gap
20	10	20	2.96	100	–	–
20	20	40	5.23	100	–	–
20	30	60	10.02	100	–	–
20	40	80	13.43	100	–	–
20	50	100	18.26	100	–	–
20	60	120	32.54	91	3.69	1.92
20	70	140	42.43	88	2.69	1.90
20	80	160	49.05	74	4.18	4.11
20	90	180	51.35	65	4.22	3.71
20	100	200	56.84	48	4.19	3.81
100	10	20	3.28	100	–	–
100	20	40	6.74	100	–	–
100	30	60	11.43	100	–	–
100	40	80	18.65	98	3.36	1.16
100	50	100	27.69	96	1.90	1.15
100	60	120	33.44	90	2.83	1.72
100	70	140	46.56	59	3.91	3.21
100	80	160	43.56	51	5.19	5.03
100	90	180	49.73	33	3.98	4.15
100	100	200	46.81	23	5.37	6.45

6.4 Experiment 3: large-scale instances

In this third experiment, we consider how well methods scale with larger instance sizes. We compare the compact and the decomposition formulations as the most promising exact methods, and two variants of the greedy method. For these variants, we do not enumerate all $O(n^2)$ candidate functions $\pi(\alpha)$ (see step 2 of the algorithm), but instead only use a grid to find values of $\hat{c}_k$ and $\hat{c}_l$ in the possible cost interval. That is, we consider the values $(\hat{c}_k, \hat{c}_l) \in \{i \cdot r/X, j \cdot r/X\} : i, j = 1, \dots, X, i \leq j\}$ with grid size X . Recall that only the values $\hat{c}_k \leq \hat{c}_l$ need to be considered on the respective grid. The grid size defines the number of restarts for the greedy method, with a finer grid resulting in more restarts and potentially better solutions. We test the grid sizes of 10×10 (“G10”) and 20×20 (“G20”).

We present computation time metrics in Table 5. Several interesting aspects can be noted. While the decomposition approach is slower than the compact formulation for $r = 100$, it is faster for $r = 20$. The compact formulation has a large difference between median and mean computation times on this dataset, which is also reflected in a high standard deviation

Table 4 Performance measures for the greedy heuristic

r	p	n	Median	Average	Stdev	Max	Opt	UB Gap	Max UB Gap
20	10	20	0.00	0.00	0.00	0.00	100	–	0.00
20	20	40	0.01	0.01	0.00	0.01	95	8.65	12.50
20	30	60	0.01	0.01	0.00	0.04	89	7.94	21.43
20	40	80	0.02	0.02	0.00	0.03	81	5.13	7.69
20	50	100	0.03	0.03	0.00	0.03	81	4.33	7.69
20	60	120	0.03	0.03	0.00	0.04	70	3.86	8.57
20	70	140	0.04	0.04	0.00	0.06	60	3.76	10.00
20	80	160	0.06	0.06	0.01	0.08	56	3.28	7.69
20	90	180	0.07	0.07	0.01	0.10	45	3.28	7.84
20	100	200	0.09	0.09	0.01	0.11	44	3.16	7.02
100	10	20	0.01	0.01	0.00	0.02	99	8.57	8.57
100	20	40	0.12	0.12	0.03	0.19	92	4.66	7.87
100	30	60	0.42	0.43	0.08	0.66	87	3.64	7.00
100	40	80	0.95	0.95	0.18	1.57	81	2.93	8.40
100	50	100	1.64	1.65	0.26	2.35	74	2.07	5.83
100	60	120	2.44	2.43	0.40	3.79	55	2.81	6.67
100	70	140	3.50	3.54	0.48	5.06	57	2.11	6.25
100	80	160	4.80	4.83	0.68	6.63	57	2.19	5.88
100	90	180	6.10	6.11	0.82	8.23	40	1.65	6.19
100	100	200	7.56	7.59	0.95	9.78	38	1.71	5.82

and large outliers. For $n = 1000$ and $r = 20$, the maximum computation time was over 3100s for the compact model, while it is less than 4s for the decomposition model. While the decomposition performs not as strong for $r = 100$, it still gives a reliable performance without outliers. This can be explained with the observation that the number of subproblems does not increase further with n large enough, and each subproblem is relatively easy to solve, scaling roughly linear in n .

We now consider the performance of the greedy heuristics. They are faster than Decom for $r = 20$, but the difference is particularly pronounced for $r = 100$. Computation times show little deviation and remain below 10s for G20 and below 3s for G10. In Table 6, these computation times are complemented with insight into the quality of solutions. For $r = 20$, G10 is on average less than 5% away from optimality, while this is less than 3% for G20. The difference between G10 and G20 is less for $r = 100$, where the additional computational effort results in small advantages in solution quality. UB Gaps scale well with instance size. Heuristics show a similar average gap for any size of instance, while maximum gaps tend to decrease due to larger absolute objective values.

Overall, we find that the decomposition model is the exact solution method of choice if there are only few candidate values that need to be checked. For such instances, the compact formulation may fail. For $r = 100$, on the other hand, the compact formulation becomes more robust and gives a better performance than the decomposition model. Greedy heuristics scale well on all types of instances and require very small computation times while being consistently only a few percent away from optimality.

Table 5 Solution times

			Median				Mean			
			Compact	Decomp	G10	G20	Compact	Decomp	G10	G20
20	150	300	0.30	1.44	0.05	0.19	0.43	1.43	0.05	0.19
20	200	400	0.46	1.73	0.08	0.30	1.14	1.74	0.08	0.31
20	250	500	0.67	2.03	0.12	0.45	2.23	2.04	0.12	0.46
20	300	600	6.17	2.32	0.16	0.64	10.34	2.34	0.17	0.64
20	350	700	9.38	2.58	0.22	0.85	31.77	2.59	0.22	0.85
20	400	800	13.59	2.85	0.28	1.10	77.92	2.88	0.28	1.10
20	450	900	18.84	3.08	0.35	1.34	70.40	3.10	0.35	1.35
20	500	1000	20.02	3.33	0.43	1.66	104.70	3.34	0.43	1.65
100	150	300	3.81	201.79	0.20	0.79	3.98	203.54	0.20	0.80
100	200	400	6.00	276.50	0.35	1.37	6.38	275.09	0.35	1.37
100	250	500	8.88	340.39	0.53	2.07	9.19	339.29	0.53	2.08
100	300	600	55.97	404.46	0.76	2.94	59.12	406.51	0.76	2.97
100	350	700	78.84	476.59	1.02	3.99	87.12	476.50	1.03	4.00
100	400	800	105.77	537.40	1.29	5.01	111.47	536.07	1.31	5.08
100	450	900	148.35	594.38	1.63	6.36	155.56	595.50	1.63	6.36
100	500	1000	177.57	662.66	2.05	7.98	186.87	659.93	2.05	7.99
			Stdev				Max			
			Compact	Decomp	G10	G20	Compact	Decomp	G10	G20
20	150	300	0.45	0.09	0.00	0.02	3.12	1.60	0.06	0.23
20	200	400	1.49	0.12	0.01	0.03	7.06	2.36	0.09	0.36
20	250	500	6.68	0.11	0.01	0.03	60.21	2.26	0.14	0.54
20	300	600	16.68	0.12	0.01	0.05	105.55	2.69	0.20	0.79
20	350	700	141.27	0.11	0.01	0.05	1029.73	2.84	0.25	0.98
20	400	800	541.12	0.11	0.02	0.06	5424.92	3.17	0.33	1.27
20	450	900	214.24	0.11	0.03	0.12	1696.06	3.36	0.51	1.98
20	500	1000	406.53	0.14	0.03	0.10	3141.56	3.67	0.49	1.94
100	150	300	0.97	14.71	0.02	0.09	7.67	236.66	0.26	1.02
100	200	400	2.02	15.42	0.03	0.13	20.30	313.63	0.43	1.68
100	250	500	2.15	16.53	0.05	0.19	17.22	382.49	0.67	2.60
100	300	600	18.44	16.03	0.06	0.24	119.01	457.38	0.96	3.70
100	350	700	29.74	15.35	0.08	0.29	179.55	514.16	1.25	4.87
100	400	800	32.54	22.10	0.09	0.35	252.60	582.21	1.55	6.04
100	450	900	39.87	18.77	0.12	0.45	286.49	642.04	1.88	7.30
100	500	1000	44.18	19.86	0.14	0.54	300.63	709.33	2.40	9.32

Table 6 Solution quality

r	p	n	Opt		UB Gap		Max UB Gap	
			G10	G20	G10	G20	G10	G20
20	150	300	11	24	4.68	2.69	17.54	8.05
20	200	400	1	12	4.55	2.53	13.92	8.55
20	250	500	1	10	4.15	2.33	12.77	8.22
20	300	600	1	3	4.60	2.42	10.81	6.02
20	350	700	3	7	4.70	2.86	13.01	7.50
20	400	800	1	1	4.73	2.81	10.40	7.30
20	450	900	1	1	4.47	2.90	13.30	6.98
20	500	1000	2	2	4.80	2.89	11.06	6.40
100	150	300	6	8	2.59	2.13	10.30	10.30
100	200	400	4	4	2.87	2.29	8.87	5.20
100	250	500	0	0	2.51	2.22	7.45	6.52
100	300	600	0	0	2.41	2.20	7.19	5.93
100	350	700	0	0	2.57	2.37	5.90	4.82
100	400	800	0	0	2.64	2.33	6.82	6.35
100	450	900	0	0	2.51	2.26	8.33	5.19
100	500	1000	0	0	2.37	2.35	4.94	4.91

7 Conclusions

In this paper we have discussed a class of two-stage combinatorial optimization problems under interval uncertainty representation. We have used the maximum regret criterion to choose the best first-stage solution. The problem has different properties than the corresponding minmax regret single-stage counterpart. In particular, there is no easy characterization of a worst-case scenario for a given first-stage solution and computing its maximum regret can be NP-hard even if the deterministic problem is polynomially solvable. We have proposed a general procedure for solving the problem, which is based on a standard row and column generation technique. This method can be used to compute optimal solutions for the problems with reasonable size. Furthermore, we have provided a characterization of the problem complexity for two variants of TSTR SHORTEST PATH and proposed compact MIP formulations for TSTR SELECTION. There is a number of open problems concerning the considered approach. The computational complexity of TSTR SELECTION is open. It is also interesting to explore the complexity of more general class of matroidal problems, for which SELECTION is a special case (another important special case is the MINIMUM SPANNING TREE problem). Finally, no approximation algorithm is known for TSTR $\mathcal{P}$. We have showed that the mid-point heuristic, used in the single-stage problems, does not guarantee any approximation ratio. We conjecture that the greedy algorithm proposed may be indeed a 2-approximation one for TSTR SELECTION. Proving this (or showing a counterexample) is an interesting open problem.

Acknowledgements The second and third author were supported by the National Science Centre, Poland, Grant 2017/25/B/ST6/00486.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give

appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix

To prove Theorem 5, we first derive two lemmas.

Lemma 3 *The coefficients $\omega_i^\pi(\alpha)$, $i \in [n]$, are nonincreasing in α .*

Proof Recall that

$$\omega_i^\pi(\alpha) = C_i - \alpha + [\alpha - \bar{c}_i]_+ + \bar{\rho}_i^\pi(\alpha) - \underline{\rho}_i^\pi(\alpha)$$

where

$$\begin{aligned}\underline{\rho}_i^\pi(\alpha) &= \max\{0, \pi(\alpha) - C_i, \pi(\alpha) - \underline{c}_i - [\alpha - \underline{c}_i]_+ + [\alpha - \bar{c}_i]_+\} \\ \bar{\rho}_i^\pi(\alpha) &= \max\{0, \pi(\alpha) - C_i, \pi(\alpha) - \underline{c}_i\}\end{aligned}$$

and

$$\pi(\alpha) = \max\{\hat{c}_k, \min\{\alpha, \hat{c}_l\}\}$$

for some fixed $\hat{c}_k \in \mathcal{A}$ and $\hat{c}_l \in \mathcal{B}$, $\hat{c}_k \leq \hat{c}_l$. We consider the following cases:

1. $\alpha \leq \hat{c}_k$: In this case $\pi(\alpha) = \hat{c}_k$ and $\bar{\rho}_i^\pi(\alpha) = \bar{\rho}_i^\pi$ is constant. Hence

$$\omega_i^\pi(\alpha) = \begin{cases} C_i - \alpha + \bar{\rho}_i^\pi - \max\{0, \hat{c}_k - C_i, \hat{c}_k - \underline{c}_i\} & \text{if } \alpha \leq \underline{c}_i \\ C_i - \alpha + \bar{\rho}_i^\pi - \max\{0, \hat{c}_k - C_i, \hat{c}_k - \alpha\} & \text{if } \underline{c}_i < \alpha < \bar{c}_i \\ C_i - \bar{c}_i + \bar{\rho}_i^\pi - \max\{0, \hat{c}_k - C_i, \hat{c}_k - \bar{c}_i\} & \text{if } \bar{c}_i \leq \alpha \end{cases}$$

and $\omega_i^\pi(\alpha)$ is nonincreasing in $[0, \hat{c}_k]$

2. $\alpha \geq \hat{c}_l$: This case is analogous to the case 1 with $\pi(\alpha) = \hat{c}_l$ and constant $\bar{\rho}_i^\pi(\alpha)$, so $\omega_i^\pi(\alpha)$ is nonincreasing in $[\hat{c}_l, \infty)$
3. $\hat{c}_k \leq \alpha \leq \hat{c}_l$: Then $\pi(\alpha) = \alpha$ and we distinguish further:

- (a) $C_i \leq \underline{c}_i \leq \bar{c}_i$: Then

$$\bar{\rho}_i^\pi(\alpha) - \underline{\rho}_i^\pi(\alpha) = [\alpha - C_i]_+ - [\alpha - C_i]_+ = 0$$

and $\omega_i^\pi(\alpha) = C_i - \alpha + [\alpha - \bar{c}_i]_+$ is nonincreasing in $[\hat{c}_k, \hat{c}_l]$.

- (b) $\underline{c}_i \leq C_i \leq \bar{c}_i$: Then

$$\bar{\rho}_i^\pi(\alpha) - \underline{\rho}_i^\pi(\alpha) = [\alpha - \underline{c}_i]_+ - [\alpha - C_i]_+$$

and therefore

$$\begin{aligned}\omega_i^\pi(\alpha) &= C_i - \alpha + [\alpha - \bar{c}_i]_+ + [\alpha - \underline{c}_i]_+ - [\alpha - C_i]_+ \\ &= \begin{cases} C_i - \alpha & \text{if } \alpha \leq \underline{c}_i \\ C_i - \underline{c}_i & \text{if } \underline{c}_i < \alpha \leq C_i \\ C_i - \underline{c}_i - \alpha + C_i & \text{if } C_i < \alpha \leq \bar{c}_i \\ C_i - \bar{c}_i - \underline{c}_i + C_i & \text{if } \bar{c}_i < \alpha \end{cases}\end{aligned}$$

and $\omega_i^\pi(\alpha)$ is nonincreasing in $[\hat{c}_k, \hat{c}_l]$.

(c) $\underline{c}_i \leq \bar{c}_i \leq C_i$: Then

$$\begin{aligned}\omega_i^\pi(\alpha) &= C_i - \alpha + [\alpha - \bar{c}_i]_+ + [\alpha - \underline{c}_i]_+ - [\alpha - \underline{c}_i - [\alpha - \underline{c}_i]_+] + [\alpha - \bar{c}_i]_+ + \\ &= \begin{cases} C_i - \alpha & \text{if } \alpha \leq \underline{c}_i \\ C_i - \underline{c}_i & \text{if } \underline{c}_i < \alpha < \bar{c}_i \\ C_i + \underline{c}_i & \text{if } \bar{c}_i \leq \alpha \end{cases}\end{aligned}$$

and $\omega_i^\pi(\alpha)$ is nonincreasing in $[\hat{c}_k, \hat{c}_l]$. $\square$

Lemma 4 For any functions $f, g, h : [a, b] \rightarrow \mathbb{R}$, attaining a maximum in $[a, b]$, where g and h are nonincreasing in $[a, b]$, it holds that

$$\max_u (f(u) + g(u)) + \max_u (f(u) + h(u)) \leq \max_u f(u) + \max_u (f(u) + g(u) + h(u)).$$

Proof Let u_1, u_2, u_3 and u_4 be such that

$$\begin{aligned}\max_u (f(u) + g(u)) &= f(u_1) + g(u_1) \\ \max_u (f(u) + h(u)) &= f(u_2) + h(u_2) \\ \max_u (f(u)) &= f(u_3) \\ \max_u (f(u) + g(u) + h(u)) &= f(u_4) + g(u_4) + h(u_4).\end{aligned}$$

We can assume without loss of generality that $u_2 \leq u_1$ (otherwise, we exchange h and g). Then $g(u_2) \geq g(u_1)$ by the monotonicity of g , $f(u_1) \leq f(u_3)$, because u_3 maximizes f and $f(u_1) + g(u_1) + f(u_2) + h(u_2) \leq f(u_3) + g(u_2) + f(u_2) + h(u_2) \leq f(u_3) + g(u_4) + f(u_4) + h(u_4)$. $\square$

We are now in the position to prove that F^π is indeed supermodular.

Proof of Theorem 5. Let any $X \subseteq Y \subseteq [n]$ and $j \in [n] \setminus Y$ be given such that $|Y| + 1 \leq p$. We need to show that $F^\pi(X \cup \{j\}) - F^\pi(X) \leq F^\pi(Y \cup \{j\}) - F^\pi(Y)$. This is the case if and only if

$$\begin{aligned}\max_\alpha \left(v^\pi(\alpha) + \sum_{i \in X} \omega_i^\pi(\alpha) + \omega_j^\pi(\alpha) \right) - \max_\alpha \left(v^\pi(\alpha) + \sum_{i \in X} \omega_i^\pi(\alpha) \right) \\ \leq \max_\alpha \left(v^\pi(\alpha) + \sum_{i \in Y} \omega_i^\pi(\alpha) + \omega_j^\pi(\alpha) \right) - \max_\alpha \left(v^\pi(\alpha) + \sum_{i \in Y} \omega_i^\pi(\alpha) \right).\end{aligned}$$

Fix $f(\alpha) = v^\pi(\alpha) + \sum_{i \in X} \omega_i^\pi(\alpha)$, $g(\alpha) = \omega_j^\pi(\alpha)$ and $h(\alpha) = \sum_{i \in Y \setminus X} \omega_i^\pi(\alpha)$. The inequality can be rewritten equivalently as

$$\max_\alpha (f(\alpha) + g(\alpha)) - \max_\alpha f(\alpha) \leq \max_\alpha (f(\alpha) + h(\alpha) + g(\alpha)) - \max_\alpha (f(\alpha) + h(\alpha)).$$

According to Lemma 3, the functions $g(\alpha)$ and $h(\alpha)$ are nonincreasing. Applying Lemma 4, the theorem thus follows. $\square$

References

- Aissi, H., Bazgan, C., & Vanderpooten, D. (2005). Complexity of the min–max and min–max regret assignment problems. *Operations Research Letters*, 33, 634–640.
- Aissi, H., Bazgan, C., & Vanderpooten, D. (2008). Complexity of the min–max (regret) versions of min cut problems. *Discrete Optimization*, 5, 66–73.
- Aissi, H., Bazgan, C., & Vanderpooten, D. (2009). Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197, 427–438.
- Aron, I. D., & van Hentenryck, P. (2004). On the complexity of the robust spanning tree problem with interval data. *Operations Research Letters*, 32, 36–40.
- Averbakh, I. (2001). On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, 90, 263–272.
- Averbakh, I., & Lebedev, V. (2004). Interval data minmax regret network optimization problems. *Discrete Applied Mathematics*, 138, 289–301.
- Chassein, A., Goerigk, M., Kasperski, A., & ZIELIŃSKI, P. (2018). On recoverable and two-stage robust selection problems with budgeted uncertainty. *European Journal of Operational Research*, 265, 423–436.
- Chassein, A. B., & Goerigk, M. (2015). A new bound for the midpoint solution in minmax regret optimization with an application to the robust shortest path problem. *European Journal of Operational Research*, 244(3), 739–747.
- Conde, E. (2004). An improved algorithm for selecting p items with uncertain returns according to the minmax regret criterion. *Mathematical Programming*, 100, 345–353.
- Conde, E. (2010). A 2-approximation for minmax regret problems via a mid-point scenario optimal solution. *Operations Research Letters*, 38, 326–327.
- Cormen, T., Leiserson, C., & Rivest, R. (1990). *Introduction to algorithms*. Cambridge: MIT Press.
- Crema, A. (2020). Min max min robust (relative) regret combinatorial optimization. *Mathematical Methods of Operations Research*, <https://doi.org/10.1007/s00186-020-00712-y>.
- Doerr, B. (2013). Improved approximation algorithms for the Min–Max selecting Items problem. *Information Processing Letters*, 113, 747–749.
- Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability. A guide to the theory of NP-completeness*. New York: W. H. Freeman and Company.
- Kall, P., & Mayer, J. (2005). *Stochastic linear programming. Models, theory and computation*. Berlin: Springer.
- Karaşan, O. E., Pınar, M. Ç., Yaman, H. (2001). *The robust shortest path problem with interval data*. Technical report, Bilkent University, Ankara.
- Kasperski, A., Kurpisz, A., & ZIELIŃSKI, P. (2013). Approximating the min–max (regret) selecting items problem. *Information Processing Letters*, 113, 23–29.
- Kasperski, A., & ZIELIŃSKI, P. (2006a). An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97, 177–180.
- Kasperski, A., & ZIELIŃSKI, P. (2006b). The robust shortest path problem in series-parallel multidigraphs with interval data. *Operations Research Letters*, 34, 69–76.
- Kasperski, A., & ZIELIŃSKI, P. (2007). On the existence of an FPTAS for minmax regret combinatorial optimization problems with interval data. *Operations Research Letters*, 35, 525–532.
- Kasperski, A., & ZIELIŃSKI, P. (2011). On the approximability of robust spanning problems. *Theoretical Computer Science*, 412, 365–374.
- Kasperski, A., ZIELIŃSKI, P. (2016). Robust discrete optimization under discrete and interval uncertainty: A survey. In M. Doumpos, C. Zopounidis & E. Grigoroudis (Eds.), *Robustness analysis in decision aiding, optimization, and analytics* (pp. 113–143). Springer, Berlin.
- Kasperski, A., & ZIELIŃSKI, P. (2017). Robust recoverable and two-stage selection problems. *Discrete Applied Mathematics*, 233, 52–64.
- Katriel, I., Kenyon-Mathieu, C., & Upfal, E. (2008). Commitment under uncertainty: Two-stage matching problems. *Theoretical Computer Science*, 408, 213–223.
- Kouvelis, P., & Yu, G. (1997). *Robust discrete optimization and its applications*. Dordrecht: Kluwer Academic Publishers.
- Montemanni, R. (2006). A Benders decomposition approach for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 174, 1479–1490.
- Montemanni, R., & Gambardella, L. M. (2004). An exact algorithm for the robust shortest path problem with interval data. *Computers and Operations Research*, 31, 1667–1680.
- Montemanni, R., & Gambardella, L. M. (2005). A branch and bound algorithm for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 161, 771–779.
- Montemanni, R., Gambardella, L. M., & Donati, A. V. (2004). A branch and bound algorithm for the robust shortest path problem with interval data. *Operations Research Letters*, 32, 225–232.

- Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Mineola: Dover Publications Inc.
- Pereira, J., & Averbakh, I. (2011). Exact and heuristic algorithms for the interval data robust assignment problem. *Computers and Operations Research*, 38, 1153–1163.
- Pereira, J., & Averbakh, I. (2013). The robust set covering problem with interval data. *Annals of Operations Research*, 207, 217–235.
- Poursoltani, M., Delage, E. (2019) *Adjustable robust optimization reformulations of two-stage worst-case regret minimization problems*. Available at Optimization Online.
- Savage, L. J. (1951). The theory of statistical decision. *Journal of the American Statistical Association*, 46, 55–67.
- Yaman, H., Karaşan, O. E., & Pınar, M. Ç. (2001). The robust spanning tree problem with interval data. *Operations Research Letters*, 29(1), 31–40.
- Zeng, B., & Zhao, L. (2013). Solving two-stage robust optimization problems using a column and constraint generation method. *Operation Research Letters*, 41, 457–461.
- Zieliński, P. (2004). The computational complexity of the relative robust shortest path problem with interval data. *European Journal of Operational Research*, 158, 570–576.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.