Minimizing maximal regret in discrete optimization problems with interval data

Adam Kasperski[†,*], Przemysław Kobylański[‡], Michał Kulej[†] and Paweł Zieliński[‡]

[†]Institute of Industrial Engineering and Management

E-mail: {adam.kasperski,michal.kulej}@pwr.wroc.pl

[‡]Institute of Mathematics

E-mail: {przemko,pziel}@im.pwr.wroc.pl

Wrocław University of Technology,

ul.Wybrzeże Wyspiańskiego 27,

50-372 Wrocław, Poland

Abstract

The paper deals with the interval data discrete optimization problems. In order to calculate the optimal solution the maximal regret criterion is adopted. This type of problems has attracted considerable attention in the recent literature. The aim of this paper is to present the state of the art as well as some open questions connected with the considered approach.

Keywords: Robust discrete optimization, Maximal regret, Interval data

1 Introduction

In the classical optimization problems all data (parameters) like costs, lengths, processing times *etc.* are assumed to be precisely known. However, this assumption may be often a serious restriction since in many real - world processes the exact values of the parameters are not known in advance. A good example is a traveling time between two cities in a communication network or a processing time of a task in scheduling

[*]Corresponding author

problem. One of the simplest method of modeling the imprecision is to define the data (parameters) as closed intervals. It is assumed that the value of each parameter may fall within a given range independently of the values taken on by the other parameters. Therefore, it is clear that an additional criterion is then required to calculate the optimal solution.

In [21] Kouvelis and Yu studied the framework of so-called *robust approach* to modeling imprecision in optimization. In general, the robust approach consists in minimizing the worst case performance of a solution over all possible values of the parameters (called *scenarios*). One of the robust criterion presented in [21] is the *maximal regret* (called also the *robust deviation*). The maximal regret is a pessimistic measure of performance and it expresses the maximal possible deviation of a given solution from optimum. Thus, the obtained optimal solution can be viewed as the most stable and the most resistant to deteriorations. We will refer to the problem of minimizing the maximal regret in optimization problem as the *robust optimization problem*.

The maximal regret criterion was first applied to linear programming. Inuiguchii and Sakawa [13] studied the general robust linear programs with interval objective function coefficients. Their solution method was based on an iterative relaxation procedure which is however exponential in the size of the problem. Mausser and Laguna [25] proposed a heuristic algorithm for that problem. Finally, Averbakh and Lebedev [7] proved that the robust linear programming problem is strongly $\mathcal{NP}$-hard. In fact, they proved that even the calculation of the maximal regret for a given solution of linear program is strongly $\mathcal{NP}$-hard.

In this paper, we focus on a certain class of the robust discrete optimization problems, which has attracted considerable attention recently. This class is defined in Section 2 and it contains such basic problems like: the minimal spanning tree, the shortest path and the assignment problems. Sections 3-6 provide a detailed exposition of these problems. Our purpose is to present the state of the art as well as some open questions connected with these problems. Unfortunately most of them turned out to be $\mathcal{NP}$-hard (the only nontrivial problem which is known to be polynomially solvable is given in Section 3). Section 7 discusses a 2-approximation algorithm which can be applied to all of the considered problems. Some computational experiments are shown which indicate that this algorithm may find quite good solutions in practical applications.

The maximal regret criterion was also applied to scheduling. For

a comprehensive review of the robust scheduling problems, we refer the reader to [15]. Some results in this field can be also found in [21] and [16]. In this paper, we do not consider the robust scheduling problems, although they are important part of the minmax regret discrete optimization.

2 Problem formulation

Let $E = \{e_1, e_2, ..., e_n\}$ be a finite set, $|E| = n$, and let $\Phi \subseteq 2^E$ be a set of subsets of E. Set Φ is called the set of *feasible solutions*. For every element $e \in E$, there is given an interval $\tilde{c}_e = [\underline{c}_e, \overline{c}_e]$, which expresses a range of possible values of the cost. A vector $S = \{c_e^S \in [\underline{c}_e, \overline{c}_e] : e \in E\}$ that represents a particular assignment of costs c_e^S to elements $e \in E$ is called *scenario*. We will denote by Γ the set of all the scenarios, i.e. Γ is the Cartesian product of the corresponding intervals $\tilde{c}_e$, $e \in E$. For a given solution $X \in \Phi$, we define its cost under a fixed scenario $S \in \Gamma$ as follows

$$F(X, S) = \sum_{e \in X} c_e^S. \tag{1}$$

We will denote by $F^*(S)$ the value of the cost of the optimal solution under scenario S,

$$F^*(S) = \min_{X \in \Phi} F(X, S), \ S \in \Gamma. \tag{2}$$

If scenario S is fixed, then problem (2) is the *classical discrete optimization problem.* Let us define

$$Z(X) = \max_{S \in \Gamma} \{F(X, S) - F^*(S)\}. \tag{3}$$

The value of $Z(X)$ is called the *maximal regret* for X. Scenario S which maximizes the right hand side of (3) is called the *worst case scenario* for X.

In this paper we focus on the following *minmax regret (robust) discrete optimization problem*

$$\text{ROB:} \quad \min_{X \in \Phi} Z(X).$$

So, we seek a feasible solution, for which the maximal regret is minimal. Problem ROB is a generalization of the classical discrete optimization problem (the classical problem boils down to determining the optimal

solution for a single scenario). It means that problem ROB is at least as hard as the corresponding classical discrete optimization problem. In particular, $\mathcal{NP}$-hardness of the classical problem implies $\mathcal{NP}$-hardness of the robust one.

The following proposition characterizes the worst case scenario for a given solution $X \in \Phi$.

Property 1 ([19]). *Given a solution $X \in \Phi$, the worst case scenario S^X for X is the one where elements $e \in X$ have costs $\overline{c}_e$ and all the other elements have costs $\underline{c}_e$, i.e. $c_e^{S^X} = \overline{c}_e$, if $e \in E$; and $c_e^{S^X} = \underline{c}_e$, if $e \in E \setminus X$.*

Making use of Property 1, we can express the maximal regret of a given solution $X \in \Phi$ in the following way

$$Z(X) = F(X, S^X) - F^*(S^X). \tag{4}$$

Note that the maximal regret of a given solution X can be calculated in polynomial time if the corresponding classical optimization problem, being a part of formula (4), is polynomially solvable. Such a situation is much better than one, for instance, in the robust linear programming, where even calculation of the maximal regret turns out to be strongly $\mathcal{NP}$-hard (see [7]).

It is worth while discussing here another important result. In this paper, we only study problems with cost function $F(X, S)$ being the sum of costs (see (1)). But this cost function can be replaced with a *bottleneck function* of the form $F(X, S) = \max_{e \in X}\{c_e^S\}$. In such a situation a general result can be shown. Suppose that the classical discrete optimization problem can be solved in $\mathcal{M}(n)$ time. Then it can be proved [4] that the corresponding robust problem with the bottleneck cost function can be solved in $\mathcal{O}(n\mathcal{M}(n))$ time. It is worth noting that if the classical problem is polynomially solvable then the corresponding robust problem is polynomial solvable as well. This property does not hold if the cost function is of the form (1).

3 The robust selecting items problem

In this section, we present a minmax regret (robust) version of the *problem of selecting p items of the minimum total cost* (RSI for short). This is the only nontrivial problem ROB, which is known to be polynomially solvable. Suppose that E, $|E| = n$, is the set of items and we must select

exactly p of them. Every item $e \in E$ has an interval cost $[\underline{c}_e, \overline{c}_e]$. The set of the feasible solutions is defined as follows:

$$\Phi = \{X \subseteq E : \ |X| = p\}.$$

We seek an alternative X, that minimizes the maximal regret. The corresponding classical problem can be solved in polynomial time by sorting all the items in nondecreasing order of their costs and then by selecting first p items.

Let us illustrate problem RSI by an example. Suppose that $E = \{1, 2, 3, 4\}$ consists of four items with the interval costs as follows: $\tilde{c}_1 = [1, 7]$, $\tilde{c}_2 = [2, 4]$, $\tilde{c}_3 = [2, 3]$, $\tilde{c}_4 = [3, 6]$. It is required to select exactly two items ($p = 2$) that minimize the maximal regret. Suppose that the selected alternative is $X = \{2, 3\}$. Then, according to Property 1, the worst case scenario S^X is the following one: $c_1^{S^X} = 1$, $c_2^{S^X} = 4$, $c_3^{S^X} = 3$, $c_4^{S^X} = 3$. The cost of the optimal solution for scenario S^X equals 4 (we select items 1 and 4) and the maximal regret $Z(X) = 3$. One can easily check that X is the optimal robust solution to this problem.

The first polynomial algorithm for problem RSI, with running time $\mathcal{O}(\min\{p, n-p\}^2 n)$, was proposed by Averbakh [5]. Conde [10] has recently improved this result by developing an algorithm that runs in $\mathcal{O}(\min\{p, n-p\}n)$ time. He has shown that the robust selecting items problem is equivalent to a certain nonlinear optimization problem with a scalar objective function. That nonlinear problem can be efficiently solved. For details we refer the reader to [10].

4 The robust spanning tree problem

We now discuss the minmax regret (robust) version of the *minimum spanning tree problem* (RST for short). Suppose that E is the set of edges of a given undirected graph $G = (V, E)$, $|V| = n$, $|E| = m$. An interval cost is associated with every edge of G. The set Φ contains all spanning trees of G. We seek a spanning tree for which the maximal regret is minimal. A sample problem is presented in Figure 1.

The computational complexity of problem RST is known. It turned out to be computationally intractable, in contrast to the classical minimum spanning tree problem, which can be solved in polynomial time, for instance, by Kruskal's algorithm [1]. Aron and Van Hentenryck [2], Averbakh and Lebedev [6] proved independently that problem RST is strongly $\mathcal{NP}$-hard even if all edges of G take costs from the same interval

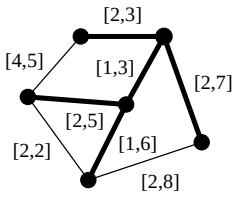


Figure 1: A graph G and a sample spanning tree T with the maximal regret $Z(T) = 10$ (the tree is in bold edges).

$[0, 1]$. It turns out that for such a cost structure RST is equivalent to the problem of determining the *central spanning tree* in G. A central spanning tree is the tree in G whose removal disconnects G into the largest number of connected components. The determination of a central spanning tree is strongly $\mathcal{NP}$-hard [8], which immediately implies that RST is strongly $\mathcal{NP}$-hard. Aron and Van Hentenryck [2] also showed that RST remains strongly $\mathcal{NP}$-hard even in the case of complete graphs.

The robust spanning tree problem turned out to be computationally intractable in general graphs. But nothing is known about the complexity of the problem for some special classes of graphs. For example, it is still the open question whether RST remains $\mathcal{NP}$-hard in planar graphs or in graphs with bounded vertex degrees. Finding an answer to these questions is an interesting subject of further research.

In the literature, there are some attempts to solve problem RST. Yaman *et al.* [27] proposed a method for solving RST based on a mixed integer programming formulation (MIP). They introduced the concepts of so-called *weak edge* and *strong edge* used for preprocessing graph G prior to solution of the robust spanning tree problem by MIP approach. An edge $e \in E$ is *weak edge* if it lies on a minimum spanning tree for some scenario (a realization of the edge costs). An edge $e \in E$ is *strong edge* if it lies on a minimum spanning tree for all scenarios. Yaman *et al.* [27] showed that a robust spanning tree, i.e. a spanning tree that minimizes the maximal regret, must be composed entirely of the weak edges. This means that non-weak edges cannot be a part of a robust spanning tree and they may be removed from G without violating the optimal solution. From the other hand, under the assumption that all the costs are non-degenerate (i.e. $\overline{c}_e > \underline{c}_e$, $e \in E$), there exists a robust spanning tree which uses every strong edge in graph G. Thus, the strong edges in G may be automatically added to a robust spanning tree.

Moreover, all the weak and strong edges in graph G can be efficiently detected in $\mathcal{O}(m^2 \log m)$ time by a slightly modified version of Kruskal's algorithm [27]. Therefore, these results can be used for preprocessing MIP model. The model contains (among others) binary variables x_e, $e \in E$, which indicates whether edge e is a part of a robust spanning tree. We may set $x_e = 0$ for each non-weak edge and $x_e = 1$ for each strong edge. Then MIP model may be solved by a standard software. Yaman *et al.* used the `CPLEX 6.5.1` MIP solver to solve the problem. They showed that the preprocessing considerably speeds up the computation of a robust spanning tree.

Another method for solving RST was proposed by Aron and Van Hentenryck [3]. The algorithm constructed in [3] is based on the constraint satisfaction approach (`CS`). It uses a lower bound for the problem and a method of pruning suboptimal solutions. The concept of weak edges is also applied and the algorithm can detect all weak edges in $\mathcal{O}(m \log m)$ time, improving the results of Yaman *et al.* [27].

Montemanni and Gambardella [24] have recently constructed a branch and bound algorithm for RST. From an algorithmic point of view that algorithm is similar to `CS` algorithm proposed in [3]. Both share the same lower bound. However, the branch and bound algorithm has stronger preprocessing, stronger reduction rules and more efficient branching strategy.

We now compare the three methods for solving RST by showing some computational results. The results come from [3], [27] and [24]. All the experiments are performed there on complete graphs with n vertices, where $n \in \{10, 15, 20, 25\}$. The cost intervals are generated randomly and they are included in the interval $[0, 40]$ (see [27] for details). The results of experiments are presented in Table 1.

n	MIP (`CPLEX 6.5.1`)	`CS`	`B and B`
10	1.96	0.28	0.04
15	33.09	4.60	1.84
20	693.88	69.95	56.46
25	2027.6	610.45	484.24

Table 1: The results of the experiments (the average computational times in CPU seconds for different values of n) [24].

As it can be seen from the experimental results, the branch and bound algorithm (`B and B`) is the best of all three. The MIP formu-

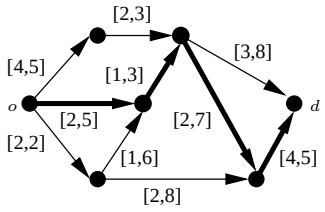


Figure 2: A graph G and a sample path p with the maximal regret $Z(p) = 12$ (the path is in bold arcs).

lation is efficient for smaller graphs ($n < 25$). Algorithm `CS` was also tested for graphs with $n = 30, 40$ and the average computational time for $n = 40$ was about 4 hours (see [3]). Hence, it may be concluded that the presented methods are efficient for not very large instances of RST (up to 40 vertices) and thus the development of more efficient exact algorithms is an important subject of further research. In Section 7, we will present a polynomial approximation algorithm, which applied to the tested instances of RST is always within very few percentage points off the optimum.

5 The robust shortest path problem

We now consider the minmax regret (robust) version of the *shortest path problem* (RSP for short). Suppose that E is the set of arcs of a given directed graph $G = (V, A)$ (that is $E = A$). An interval cost is associated with each arc. Two vertices $o \in V$ and $d \in V$ are distinguished as the *origin* and *destination* vertex, respectively. The set Φ consists of all paths from o to d in G. We seek a path from o to d for which the maximal regret is minimal. A sample problem is shown in Figure 2. The classical shortest path problem is well known and it can be solved in polynomial time by Dijkstra's algorithm [11] in a graph, in which there are no negative cycles.

The computational complexity of RSP is known. Zieliński [28] proved that the problem is $\mathcal{NP}$-hard even when a graph is restricted to be directed, acyclic and planar with vertex degrees at most three (the degree of vertex $v \in V$ is the sum of the number of its incoming and outgoing arcs). This result is also valid if the graph is undirected (the proof goes almost without any modifications). At the same time Averbakh and Lebedev [6] proved that the problem is strongly $\mathcal{NP}$-hard for a general

directed (also for undirected) graph even if the bounds of all the cost intervals belong to $\{0,1\}$. Finally, Kasperski and Zieliński [17] proved that RSP remains $\mathcal{NP}$-hard even in series-parallel graphs with vertex degrees at most three. This result shows that RSP is computationally intractable even for a very restricted class of graphs. However, for the general series-parallel graphs the problem is not strongly $\mathcal{NP}$-hard, since in [17] a pseudopolynomial algorithm for this class of graphs was constructed.

Problem RSP was first studied by Karasan *et al.* [14]. They proposed a MIP model for the problem. They also used a concept of *weak arc*, which is analogous to the concept of weak edge in problem RST (see Section 4). An arc is said to be *weak arc* if it belongs to a shortest path in some scenario (realization of the arc costs). They observed that a robust shortest path is composed of weak arcs, so all the non-weak arcs can be removed from graph G without violating the optimal solution. Unfortunately, the problem of detecting weak (and non-weak) arcs is $\mathcal{NP}$-hard even in planar graphs [9]. So the preprocessing based on removing the non-weak arcs cannot be efficient in general graphs. Karasan *et al.* [14] investigated problem RSP in *layered graphs*. A layered graph has the following properties: the vertex set V can be partitioned into disjoint subsets $V = \{o\} \cup V_1, \ldots, V_k \cup \{d\}$; the arcs exist only from o to V_1, from V_k to d and from V_i to V_{i+1} for $i = 1, \ldots, k-1$. A number $w = \max\{|V_i| : \ i = 1, \ldots, k\}$ is called the *width* of graph G. It is clear that, every acyclic graph can be converted into a layered graph by adding dummy vertices and arcs. Karasan *et al.* studied the problem of detecting the non-weak arcs in layered graphs. They gave only a necessary condition for an arc to be weak and proposed a polynomial algorithm, which detects some (but not all) non-weak arcs. Then, the preprocessed instance was solved by means of the MIP model. One can observe that this approach is efficient for graphs with small width and it is restricted only to acyclic graphs.

An exact algorithm for the general problem RSP was proposed by Montemanni and Gambardella [22]. We briefly describe its main properties. Let S^u denote scenario in which all the arc costs are set to their upper bounds, i.e. $c_a^{S^u} = \overline{c}_a$, $a \in A$. Assume that all the paths in Φ are ranked in nondecreasing order of their lengths under scenario S^u. Thus path p_1 is the shortest path under S^u and $p_{|\Phi|}$ is the longest path under S^u. The algorithm proposed by Montemanni and Gambardella retrieves paths $p_1, \ldots, p_K$ for some fixed value of K, $1 \leq K \leq |\Phi|$. It is terminated when the optimality condition, proposed in [22], is fulfilled

(an exact solution is determined) or when a given number of paths K has been examined (a heuristic solution is obtained). The algorithm is based on the intuitive conjecture that the path ranking under scenario S^u is also a good ranking in terms of the maximal regret. We will show in Section 7 that this conjecture is false in a general case and path p_1 can be arbitrarily far from the optimum. But experimental results, presented in [22], show that the average performance of the algorithm is quite good.

Montemanni *et al.* [23] constructed a branch algorithm for RSP. That algorithm can be seen as an improvement of the method proposed in [22]. It is based on a lower bound, a branching strategy and some reduction rules which speed up computations.

Let us now present some selected results of experiments. The results come from [23], where the following three types of graphs were considered:

- *Random graphs.* $R-n-c-\delta$, where n is the number of vertices, δ is approximate arc density, i.e $|A| \approx \delta n(n-1)$, costs are randomly generated in such a way that $\overline{c}_a \in [0, c]$ and $\underline{c}_a \in [0, \overline{c}_a]$ for all $a \in A$.

- *Karasan graphs.* $K-n-c-d-w$ is the family of layered graphs, where n is the number of vertices, w is the width of the graph and c and d determine the cost structure as follows: for every arc $a \in A$ select randomly $c^* \in [0, c]$; then select randomly $\underline{c}_a \in [(1-d)c^*, (1+d)c^*]$ and $\overline{c}_a \in [\underline{c}_a, (1+d)c^*]$ (see [14] for more detailed description of this family of graphs).

- *Stutgart.* This graph represents the real road network of the Stuttgart area (Germany), and its interval costs are realistic. The graph has 2490 vertices and 16 153 arcs.

The results of experiments are presented in Table 2.

In Table 2 MIP model, solved by `CPLEX`, is compared to the branch and bound algorithm (`B and B`). One can observe that the branch and bound algorithm is more effective for the problems based on random graphs, although the performance of `CPLEX` is also acceptable. Algorithm `B and B` also works better for the real graph *Stuttgart*. `CPLEX` is faster for *Karasan graphs*. Montemanni *et al.* [23] tried to explain this not so good performance by observing that all the paths in *Karasan graphs* consist of large number of arcs and thus there is a huge number of alternative paths from o to d.

Graph	MIP (CPLEX 6.0)	B and B
R-500-100-0.1	4.633	1.481
R-500-100-0.001	0.578	0.599
R-500-100-0.1	14.365	2.388
R-100-100-0.01	0.079	0.046
R-900-100-0.01	25.161	5.244
R-500-10-0.01	5.033	0.553
R-500-1000-0.01	5.863	3.069
K-30-20-0.9-2	0.016	0.025
K-60-20-0.9-2	0.088	7.085
Stuttgart	24.611	6.285

Table 2: The average computational times in CPU seconds for each set of problems [23]

The robust shortest path problem is strongly $\mathcal{NP}$-hard for general graphs. It is interesting to check whether this problem remains $\mathcal{NP}$-hard for some special classes of graphs. In graph theory there exists a class of graphs called *edge series-parallel multidigraphs* (shortly ESP). These graphs have several applications, for instance, in the analysis of electrical circuit, scheduling *etc.* Moreover, many combinatorial problems, which are hard for general graphs, become polynomially solvable for ESP.

An *edge series-parallel multidigraph* is recursively defined as follows (see [26]). A graph consisting of two vertices joined by a single arc is ESP. If G_1 and G_2 are ESP, so are the multidigraphs constructed by each of the operations:

- *parallel composition* $p(G_1, G_2)$: identify the source of G_1 with the source of G_2 and the sink of G_1 with the sink of G_2.
- *series composition* $s(G_1, G_2)$: identify the sink of G_1 with the source of G_2.

It results from this definition that each ESP is acyclic. An edge series-parallel multidigraph G is naturally associated with a rooted binary tree $\mathcal{T}$ called the *binary decomposition tree of* G. Each leaf of the tree represents an arc in G. Each internal node of $\mathcal{T}$ is marked $\mathcal{S}$ or $\mathcal{P}$ and represents the series or parallel composition in G represented by subtrees rooted at the children of the node (see Figure 3).

Kasperski and Zieliński [17] proved that problem RSP is $\mathcal{NP}$-hard even for ESP in which the vertex degrees are at most three. This neg-

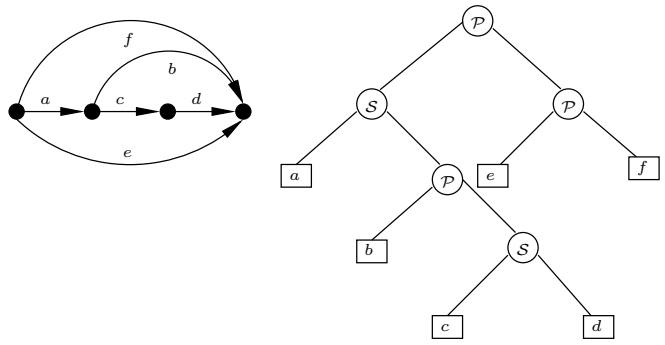
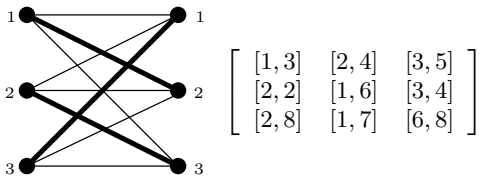


Figure 3: An ESP and its binary decomposition tree.

ative result shows that problem RSP remains $\mathcal{NP}$-hard even in a very restrictive case. However, there are also two positive results for the considered class of graphs. The preprocessing based on removing the non-weak arcs can be performed in ESP in polynomial time in contrast to the case for general planar graphs. It is the consequence of results obtained by Dubois *et al.* [12]. The second positive result is a pseudopolynomial algorithm, constructed in [17], with the complexity of $\mathcal{O}(mL_{max}^2)$, where m is the number of arcs in ESP and L_{max} is the length of the longest path from o to d under the scenario in which all the arc costs are set to their upper bounds. The algorithm determines the binary decomposition tree of a given G (this can be done in $\mathcal{O}(m)$ time [26]). Then, it performs the series and the parallel compositions storing subpaths, which can be a part of the optimal robust path. The number of the stored subpaths is bounded by the maximal values of costs in the problem. For details, we refer the reader to [17].

6 The robust assignment problem

This section deals with a minmax regret (robust) version of the *assignment problem* (RAS for short). Suppose that E is the set of edges of a given bipartite graph $G = (V, W; E)$, where the vertex sets V and W have n vertices. The costs of the edges of G are specified as intervals. The set Φ contains all *assignments (perfect matchings)* in G. The robust assignment problem consists in finding an assignment for which the maximal regret is minimal. A sample problem is shown in Figure 4. The classical assignment problem can be solved in polynomial time, for



$$\begin{bmatrix} [1,3] & [2,4] & [3,5] \\ [2,2] & [1,6] & [3,4] \\ [2,8] & [1,7] & [6,8] \end{bmatrix}$$

Figure 4: A bipartite graph G and a sample assignment A with the maximal regret $Z(A) = 13$ (the assignment is in bold edges).

instance, by Hungarian algorithm [20].

The robust assignment problem was studied by Kasperski and Zieliński [18]. This problem turned out to be $\mathcal{NP}$-hard [18]. Kasperski and Zieliński [18] proposed a MIP model for problem RAS and up to now it is the only known method for solving the problem. We present this model because it is relatively simple. It is similar in spirit to the mixed integer models for RST and RSP. Assume that all the elements in set W (and similarly in V) are numbered from 1 to n. Let us introduce binary variables $x_{ij} \in \{0,1\}$, $i = 1,\ldots,n$, $j = 1,\ldots,n$, such that $x_{ij} = 1$ if and only if element $i \in W$ is assigned to element $j \in V$. Every assignment X can be represented as a matrix $[x_{ij}]_{n\times n}$, which elements fulfill the assignment constraints. The assignment X which minimizes the maximal regret can be then obtained by means of the following MIP model:

$$\begin{array}{lll} & \sum_{i=1}^{n}\sum_{j=1}^{n}\overline{c}_{ij}x_{ij} - \sum_{i=1}^{2n} y_i \to \min & \\ 1: & \sum_{i=1}^{n} x_{ij} = 1 & \text{for} \quad j = 1,\ldots,n, \\ 2: & \sum_{j=1}^{n} x_{ij} = 1 & \text{for} \quad i = 1,\ldots,n, \\ 3: & y_i + y_{n+j} \le \overline{c}_{ij}x_{ij} + \underline{c}_{ij}(1-x_{ij}) & \text{for} \quad i,j = 1,\ldots,n, \\ 4: & x_{ij} \in \{0,1\} & \text{for} \quad i,j = 1,\ldots,n. \end{array} \tag{5}$$

Constraints 1 and 2 are the typical assignment constraints and they assure that matrix $[x_{ij}]_{n\times n}$ represents an assignment. The cost of assigning i to j under the worst case scenario S^X induced by X can be expressed as $\overline{c}_{ij}x_{ij} + \underline{c}_{ij}(1-x_{ij})$. Constraints 3 are the dual constraints to the classical assignment problem for scenario S^X. Finally, we have $F(X,S^X) = \sum_{i=1}^{n}\sum_{j=1}^{n}\overline{c}_{ij}x_{ij}$ and $F^*(S^X)$ is the maximal value of $\sum_{i=1}^{2n} y_i$. Hence, the objective function of (5) expresses the maximal regret of X.

Let us now present some results of experiments [18]. In order to solve the MIP model (5) `CPLEX 3.6.1` solver was used. The experiments

were carried out for a family of problems denoted by S_C^n, where n is the number of vertices in V (W) and C specifies the variability of the costs in the following way: for every edge $e \in E$ the value of $\underline{c}_e$ is uniformly distributed in the interval $[0, C]$ and $\overline{c}_e$ is uniformly distributed in interval $[\underline{c}_e, \underline{c}_e + C]$. The experimental results for different values of n and C are shown in Table 3.

$C \setminus n$	20	50	80	110	140	160	180	200
10	1.3	10.1	4.7	13.5	23.1	42.2	55.7	65.9
30	1.6	37.0	137.9	242.6	299.8	443.4	419.8	400.0
50	1.2	43.6	203.8	519.0	1309.7	1495.5	1305.3	1316.5
70	1.9	54.0	255.9	512.9	2029.8	2005.5	2689.6	3978.1
100	1.5	61.4	195.7	490.3	1644.9	5133.8	5178.4	8689.2
150	1.4	68.1	245.4	937.9	1604.7	3747.7	7106.9	11665.2

Table 3: The results of the experiments (the average computational times in CPU seconds for each set of problems) [18].

One can observe that MIP approach is efficient if up to 400 elements must be paired. The computational time depends on the variability of the costs, that is the greater is the value of C the longer is the computational time.

7 Approximation algorithms for problem ROB

In this section, we discuss some approximation algorithms for problem ROB. The idea of these algorithms consists in solving the classical problem corresponding to ROB for a certain scenario. The approximation algorithms will be polynomial if the classical discrete optimization problem (2) can be solved in polynomial time. Note that this is the case for all of the problems considered in Sections 3-6.

The first algorithm, denoted by `AM`, solves the classical problem (2), related to ROB, for the scenario in which the costs of the elements of E are the midpoints of their corresponding cost intervals.

ALGORITHM `AM`
for all $e \in E$ **do**
  $c_e^S \leftarrow \frac{1}{2}(\underline{c}_e + \overline{c}_e)$;
end for
$M \leftarrow \arg\min_{X \in \Phi} F(X, S) = \arg\min_{X \in \Phi} \sum_{e \in X} c_e^S$;
return M;

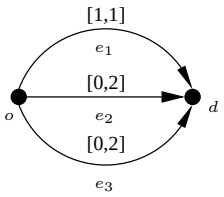


Figure 5: An RSP for which algorithm AM achieves a ratio of 2.

Kasperski and Zieliński [19] proved that the performance ratio of Algorithm AM is 2, i.e. if X^* is the optimal solution of problem ROB and M is the solution determined by Algorithm AM, then inequality $Z(M) \leq 2Z(X^*)$ holds. In particular, if $Z(X^*) = 0$ then $Z(M) = 0$ and Algorithm AM finds the optimal solution for problem ROB.

It easy to construct an example of problem ROB showing that 2 is the worst case performance ratio of algorithm AM. Consider the robust shortest path problem presented in Figure 5 (sets: $E = \{e_1, e_2, e_3\}$, $\Phi = \{\{e_1\}, \{e_2\}, \{e_3\}\}$ and the arc costs: $\tilde{c}_{e_1} = [1, 1]$, $\tilde{c}_{e_2} = [0, 2]$, $\tilde{c}_{e_3} = [0, 2]$).

The maximal regrets of all the paths (the solutions from Φ) are as follows: $Z(\{e_1\}) = 1$, $Z(\{e_2\}) = Z(\{e_3\}) = 2$. It is easy to check that algorithm AM may return each of these paths. The worst alternatives are $\{e_2\}$ or $\{e_3\}$, which give the bound of 2.

Algorithm AM can be additionally refined. Consider the following algorithm AU for problem ROB.

ALGORITHM AU
for all $e \in E$ **do**
  $c_e^S \leftarrow \overline{c}_e$;
end for
$M \leftarrow \arg\min_{X \in \Phi} F(X, S) = \arg\min_{X \in \Phi} \sum_{e \in X} c_e^S$;
return M;

Algorithm AU solves the classical problem (2), related to ROB, for scenario, in which all the costs are set to their upper bounds. Algorithm AU can be much worse than AM. In [19], it is presented an example of the robust shortest path problem for which the performance ratio of AU is unbounded. However, some experiments showed that algorithm AU often finds solutions better than AM. Based on this observation, we are thus led to the following strengthening of Algorithm AM.

ALGORITHM AMU
$M_1 \leftarrow$ Algorithm AM;
$M_2 \leftarrow$ Algorithm AU;
if $Z(M_1) < Z(M_2)$ **then**

n	AM		AU		AMU	
	worst%	average%	worst%	average%	worst%	average%
5	20.51	1.06	75.00	12.51	3.64	0.30
10	13.21	3.67	40.00	6.14	11.11	1.56
15	11.67	4.97	13.04	2.96	8.70	2.01
20	13.51	5.02	8.92	1.52	8.92	1.43

Table 4: The worst and the average deviations from the optimum for different values of n (the number of vertices in a graph) [19].

return M_1;
else
return M_2;
end if

Algorithm AMU is a combination of AM and AU. It simply returns the better solution produced by AM or AU. It is obvious that performance ratio of algorithm AMU is at most 2. Kasperski and Zieliński [19] conjecture that this performance is better, but the proof as well as a counterexample to this hypothesis are unknown.

In [19], the performance of algorithms AM, AU and AMU was evaluated experimentally. The algorithms were applied to the robust spanning tree problem (RST). They were tested on the same family of graphs as considered in [27] (see also Section 3). For each instance of RST, the MIP approach was used for obtaining the optimal solution. The optimal solutions were then compared to the solutions determined by algorithms MU, AU and AMU. The results of the experiments are presented in Table 4.

As it can be seen in Table 4, algorithm AMU performs much better than AU and AM separately. For the considered type of graphs with random edge costs, algorithm AMU is always within very few percentage points off the optimum. The worst returned solution is 11.11% from the optimum, which is much better than the theoretical worst case performance equal to 100%.

8 Conclusions

The robust discrete optimization involving the maximal regret criterion has attracted considerable attention recently. Perhaps, there are two reasons of such interest: modeling data (parameters) as intervals is the

simplest form of uncertainty representation and the maximal regret criterion is very natural and it has a clear interpretation. Moreover, the robust problem can be viewed as a direct generalization of a classical deterministic one.

In the last years a great progress in the robust discrete optimization has been made. In particular, the computational complexity of most of the basic problems has been obtained. Unfortunately, the robust problems which are polynomially solvable in the deterministic case, like the minimum spanning tree problem , the shortest path problem and the assignment problem, turned out to be $\mathcal{NP}$-hard. However, the complexity of some special cases of these problem is still unknown. For instance, it is not known whether the robust spanning tree problem is strongly $\mathcal{NP}$-hard in planar graphs. The questions like this should be the subject of further research.

The robust discrete optimization problems, presented in this paper, can be solved either by MIP formulation or by some special algorithms. MIP formulation is enough for not very large instances of the problems. Larger ones can be solved by more sophisticated methods like branch and bound. The improvement of exact algorithms is the subject further research. Alternatively, it is always possible to apply the approximation algorithms presented in Section 7, which guarantee that the determined solution will be at most 2 times worse than the optimum.

References

[1] Ahuja R. K., Magnanti T. L., Orlin J. B. (1993), *Network Flows, Theory, Algorithms and Applications*, Prentice Hall, New Jersey.

[2] Aron I. D, Van Hentenryck P. (2004), *On the complexity of the robust spanning tree problem with interval data*, Operation Research Letters **32**, 36–40.

[3] Aron I., van Hentenryck P. (2002), *A Constraint Satisfaction Approach to the Robust Spanning Tree with Interval Data*, Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence, Edmonton, Canada.

[4] Averbakh I. (2000), *Minimax regret solutions of minimax optimization problems with uncertainty*, Operation Research Letters **2**, 57–65.

[5] Averbakh I. (2001), *On the complexity of a class of combinatorial optimization problems with uncertainty*, Mathematical Programming **90**, 263–272.

[6] Averbakh I., Lebedev V. (2004), *Interval data mminmax regret network optimization problems*, Discrete Applied Mathematics **138**, 289–301.

[7] Averbakh I., Lebedev V. (2005), *On the complexity of minmax linear programming*, European Journal of Operational Research **160**, 227–231.

[8] Bezrukov S., Kaderali F., Poguntke W. (1996), *On central spanning trees of a graph*, in: Lectures Notes on Computer Sciences, Vol. 1120, Springer, Berlin, 53–58.

[9] Chanas S., Zieliński P. (2003), *On the hardness of evaluating criticality of activities in a planar network with duration intervals*, Operations Research Letters **31**, 53–59.

[10] Conde E. (2004), *An improved algorithm for selecting p items with uncertain returns according to the minimax-regret criterion*, Mathematical Programming **100**, 345–353.

[11] Dijkstra E. W. (1959), *A note on two problems in connection with graphs*, Numerische Mathematik **1**, 269–271.

[12] Dubois D., Fargier H., Galvagnon V. (2003), *On latest starting times and floats in activity networks with ill-known durations*, European Journal of Operational Research **147**, 266–280.

[13] Inuiguchi M., Sakawa M. (1997), *Minimax regret solution to linear programming problems with an interval objective function*, European Journal of Operational Research **86**, 526–536.

[14] Karasan O. E., Pinar M. C. (2002), Yaman H., *The robust shortest path problem with interval data*, Computers and Operations Research.

[15] Kasperski A. (2004), *Minimizing maximal regret in sequencing problems with interval parameters*, in: Soft Computing Foundations and Theoretical Aspects, K. Atanassov, O. Hryniewicz, J. Kacprzyk (eds.), Akademicka Oficyna Wydawnicza EXIT, Warszawa, 227–291

[16] Kasperski A. (2003), *Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion*, to appear in Operations Research Letters.

[17] Kasperski A., Zieliński P. (2004),*The robust shortest path problem in series-parallel multidigraphs with interval data*, Instytut Matematyki PWr., Wroclaw, raport serii PREPRINTY nr 017, submitted for publication in Operations Research Letters.

[18] Kasperski A., Zieliński P. (2004) *Minimizing maximal regret in linear assignment problems with interval data*, Instytut Matematyki PWr., Wroclaw, raport serii PREPRINTY nr 007, submitted for publication in Discrete Applied Mathematics.

[19] Kasperski A., Zieliński P. (2004), *An approximation algorithm for interval data minmax regret combinatorial optimization problems*, Instytut Matematyki PWr., Wroclaw, raport serii PREPRINTY nr 044, submitted for publication in Information Processing Letters.

[20] Kuhn H. W. (1955), *The Hungarian method for the assignment problem*, Naval Research Logistics Quarterly **2** 83–97.

[21] Kouvelis P., Yu G. (1997), *Robust discrete optimization and its applications*, Kluwer Academic Publishers, Boston.

[22] 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.

[23] 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**, 224–232.

[24] 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.

[25] Mausser H. E., Laguna M. (1999), *A heuristic to minimax absolute regret for linear programs with interval objective function coefficients*, European Journal of Operational Research **117**, 157–174.

[26] Valdes J., Tarjan R. E., Lawler E. L. (1982), *The recognition of series parallel digraphs*, SIAM Journal on Computing **11**, 298–313.

[27] Yaman H., Karasan O. E., Pinar M. (2001), *The robust spanning tree problem with interval data*, Operations Research Letters **29**, 31–40.

[28] Zielinski P. (2004), *The computational complexity of the relative robust shortest path problem with interval data*, European Journal of Operational Research **158**, 570-576.