

Making Robust Decisions in Discrete Optimization Problems as a Game against Nature

Adam Kasperski*

Received 26 September 2008; Accepted 7 December 2008

Abstract In this paper a discrete optimization problem under uncertainty is discussed. Solving such a problem can be seen as a game against nature. In order to choose a solution, the minmax and minmax regret criteria can be applied. In this paper an extension of the known minmax (regret) approach is proposed. It is shown how different types of uncertainty can be simultaneously taken into account. Some exact and approximation algorithms for choosing a best solution are constructed.

Keywords Discrete optimization, minmax, minmax regret, game against nature

JEL classification C61, C79

1. Introduction

Decision making under uncertainty is an important area of research in economy. If one tries to describe a given system, some parameters often appear whose values are not precisely known. This uncertainty can be seen as a feature of the nature. In one of the most popular approaches to hedging against such uncertainty, a set of all possible realizations of the parameters is specified. Every particular realization is called a *scenario* and, in the simplest case, no probability distribution in the scenario set is given. The decision making process can be seen as a zero-sum game against nature (Luce and Raiffa 1957). To guarantee a certain payoff, we may wish to make a decision that has the best performance under the worst scenario which may appear. This leads to applying well known game theoretic criteria, namely the minmax and minmax regret ones. Under the minmax criterion we choose a decision whose maximal cost over all scenarios is minimal and under the minmax regret one we choose a decision whose maximal regret (opportunity loss) over all scenarios is minimal. The minmax regret criterion was first suggested by Savage (1951).

In a wide class of decision making problems we seek a cheapest object composed of some elements of a given finite set. This class is called *discrete optimization problems*. Suppose that we explore a part of a communication network. We can model this network as a graph $G = (V, E)$, where a finite set of edges E represents roads. Every road $e \in E$ has an associated cost c_e , which may for instance express a traveling time of e . A decision consists of choosing a best path between two points in the network.

* Wrocław University of Technology, Institute of Industrial Engineering and Management, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland. Phone +48 (71) 3204205, E-mail: adam.kasperski@pwr.wroc.pl.

Under deterministic costs the cheapest path can be found by many known and efficient algorithms (see e.g. Ahuja et al. 1993). The situation, however, is more complex if precise values of the costs are not known. In this case we only have a scenario set containing more than one cost realization which may occur. Since now the total cost of a path is not known, we must use some additional criteria to make a decision. We may treat this problem as a game against nature, that is a specific player who always tries to increase the cost of our decision. We may thus apply the minmax or minmax regret criterion to choose a path.

The minmax (regret) approach to discrete optimization has attracted a considerable attention in recent decade. Many results in this area have been described in a book by Kouvelis and Yu (1997). Since the 1997's book a number of papers devoted to this approach have appeared (e.g. Aissi et al. 2005, 2007, Aron and van Hentenryck 2004, Averbakh 2001, Averbakh and Lebedev 2004, Conde 2004, Kasperski and Zieliński 2006, Yaman et al. 2001). Some surveys of recent results can be found in a paper by Aissi et al. (2008) and in a book by Kasperski (2008). In general, the problem with more than one possible cost realization turned out to be more complex to solve than its deterministic counterpart. The exact methods of obtaining a solution are based on a mixed integer programming formulation (Kouvelis and Yu 1997, Yaman et al. 2001) or a branch and bound procedure (Kouvelis and Yu 1997, Montemanni et al. 2004, 2005). There are also some approximation algorithms that can be used to obtain an approximate solution in polynomial time (Aissi et al. 2007, Kasperski and Zieliński 2006). In this paper we propose a new robust model, in which scenario set is a union of a finite number of so called *interval scenarios*. We generalize in this way the robust models discussed in literature. This new type of scenario set allows us to take into account different kinds of uncertainty, which has been treated separately in literature so far. We also focus on some exact and approximation methods of solving the constructed problems.

This paper is organized as follows. In Section 2 we briefly recall a very simple and well known minmax (regret) decision making model. We introduce some notations, which will be next extended to a problem with more complex structure. In Section 3 we generalize the minmax (regret) decision making model to the class of discrete optimization problems. We recall a known approach and we propose its extension based on a set of interval scenarios. Section 4 is devoted to some methods of solving the constructed problems. We design a mixed integer programming model, which can be solved by a standard software and, for large problems, we propose an approximation algorithm.

2. A simple minmax (regret) decision making model

In this section we recall a well known and very simple decision making model (see e.g. Luce and Raiffa 1957). In the next sections we will show how the notions and concepts introduced here can be naturally extended. We are given a finite set of elements $E = \{e_1, \dots, e_n\}$. A *decision* consists of choosing a single element from the set E . Let $\mathbf{c} = [c_{e_1}, \dots, c_{e_n}]$ be vector of nonnegative real numbers, where c_e is a cost of element

$e \in E$ under \mathbf{c} . We will use $F(e, \mathbf{c}) = c_e$ and $\hat{F}(\mathbf{c}) = \min_{e \in E} F(e, \mathbf{c})$ to denote the cost of e under \mathbf{c} and the cost of a best decision under \mathbf{c} respectively. The quantity $D(e, \mathbf{c}) = F(e, \mathbf{c}) - \hat{F}(\mathbf{c})$ is called a *regret* of decision e under \mathbf{c} and it expresses an opportunity loss when decision e is chosen under costs \mathbf{c} . In a deterministic case, where there is only one cost vector \mathbf{c} , we simply choose a decision having the smallest cost or, equivalently, whose regret equals 0.

Assume now that the cost vector is not known in advance. Instead of a single vector, there is a finite set $\Gamma = \{\mathbf{c}^1, \dots, \mathbf{c}^k\}$ of cost vectors called a *scenario set*. Every *scenario* $\mathbf{c}^j \in \Gamma$ may appear with positive but perhaps unknown probability. So, we know that exactly one cost realization from Γ will appear but it is not possible to predict which one. The problem of choosing a decision can be seen as a zero-sum game against a specific player called a nature. In this context, the scenario set Γ represents all possible states (strategies) of the nature. In order to make a decision two criteria are widely applied, namely *minmax* and *minmax regret* ones. Under the minmax criterion we chose a decision whose maximal cost over all scenarios is minimal, that is we solve problem $\min_{e \in E} \max_{\mathbf{c} \in \Gamma} F(e, \mathbf{c})$ and under the minmax regret criterion we choose a decision whose maximal regret over all scenarios is minimal, that is we solve problem $\min_{e \in E} \max_{\mathbf{c} \in \Gamma} D(e, \mathbf{c})$. We will call the decisions obtained by solving both problems a *minmax* and *minmax regret decision* respectively.

A deeper interpretation and a critical discussion on both criteria can be found in books by Kouvelis and Yu (1997) and Luce and Raiffa (1957). In general, we should apply the minmax criterion if we only wish to minimize the cost of our decision. On the other hand, the minmax regret criterion is appropriate if we have a competitor and we wish to minimize his superiority over us. In this case we may assume that our competitor always chooses a best decision under every scenario and choosing a minmax regret decision we minimize the maximal dominance of the competitor. This may be more important than simply minimizing the maximal cost.

3. Minmax (regret) discrete optimization problem

We now show how the simple decision making problem described in the previous section can be extended. Suppose that, in addition to E , we are given another set Φ that contains some subsets of E , that is $\Phi \subseteq 2^E$. The set Φ is called a set of *feasible solutions* and now decision consists of choosing a feasible solution $X \in \Phi$. Extending the notations from the previous section, we will use $F(X, \mathbf{c}) = \sum_{e \in X} F(e, \mathbf{c})$ to denote the cost of solution X under \mathbf{c} and by $\hat{F}(\mathbf{c}) = \min_{X \in \Phi} F(X, \mathbf{c})$ the cost of a best (optimal) solution under \mathbf{c} . We can express the regret of solution X under \mathbf{c} as $D(X, \mathbf{c}) = F(X, \mathbf{c}) - \hat{F}(\mathbf{c})$. A triple (E, Φ, \mathbf{c}) is called a *deterministic discrete optimization problem* and our aim is to choose a best solution under the only cost realization \mathbf{c} .

Using different descriptions of the set Φ we get different problems. In an important class of *network problems*, E is a set of edges of a given graph $G = (V, E)$ and Φ contains subsets of edges that form, for example, paths, spanning trees, perfect matchings, cuts etc. in G . If E is a set of items, then Φ may contain all subsets of items whose cardinalities are precisely p (the minimum selecting items problem). This problem can

be generalized by introducing a positive capacity p_e for every item $e \in E$ and Φ contains then all subsets of items whose total capacities do not exceed a given number P . This is well known 0-1 knapsack problem. A comprehensive review of various problems can be found for instance in books by Ahuja et al. (1993) and Papadimitriou and Steiglitz (1998).

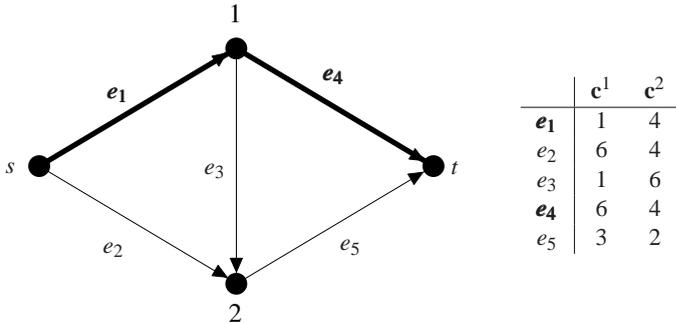
Suppose now that, similarly to the problem discussed in the previous section, we are given a finite set of scenarios $\Gamma = \{\mathbf{c}^1, \dots, \mathbf{c}^k\}$ that represent some states of the nature, that is some possible vectors of the element costs which may occur. In order to choose a solution we can use exactly the same reasoning as in Section 2. The only significant difference is that now decision set is given by Φ . So, we may seek a *minmax solution* minimizing the maximal cost or a *minmax regret solution* minimizing the maximal regret over all scenarios from Γ . These solutions can be obtained by solving the following optimization problems:

$$\text{MINMAX: } \min_{X \in \Phi} \max_{\mathbf{c} \in \Gamma} F(X, \mathbf{c}),$$

$$\text{MINMAX REGRET: } \min_{X \in \Phi} \max_{\mathbf{c} \in \Gamma} D(X, \mathbf{c}).$$

Example 1. Consider a sample problem shown in Figure 1. In this problem $E = \{e_1, \dots, e_5\}$ is a set of arcs of a given directed graph $G = (V, E)$ and Φ contains all subsets of the arcs that form paths between nodes s and t in G . So, $\Phi = \{\{e_1, e_4\}, \{e_1, e_3, e_5\}, \{e_2, e_5\}\}$ contains three decisions (paths). We have two possible scenarios, so $\Gamma = \{\mathbf{c}^1, \mathbf{c}^2\}$ and the costs under both of them are shown in the table in Figure 1.

Figure 1. A sample shortest path problem with two deterministic scenarios



It holds $\hat{F}(\mathbf{c}^1) = 5$ and $\hat{F}(\mathbf{c}^2) = 6$. In other words, the shortest path under \mathbf{c}^1 has cost 5 and the shortest path under \mathbf{c}^2 has cost 6. Consider a sample path $X = \{e_1, e_4\}$. This path has the cost 7 under scenario \mathbf{c}^1 and 8 under \mathbf{c}^2 . It also has regret equal to 2 under \mathbf{c}^1 and 2 under \mathbf{c}^2 . So, the maximal cost of X under all scenarios is 7, while its maximal regret over all scenarios is 2. It is easy to check that X is the best path under both minmax and minmax regret criteria. It is, however, optimal neither under \mathbf{c}^1 nor \mathbf{c}^2 . It is a compromise solution that has the best performance in the worst case. \square

There is one very significant difference between the discrete optimization problem and the decision problem described in Section 2. The solution space (the cardinality of Φ) in a discrete optimization problem is typically exponential in the number of elements in E . So, for real problems, we cannot find a best solution by simply exploring the whole set Φ . Unfortunately, a general efficient algorithm for computing a minmax (regret) solution probably does not exist because such problems as shortest path, minimum spanning tree, minimum assignment, minimum cut and minimum selecting items turned out to be NP-hard even for 2 scenarios (Aissi et al. 2005, Kouvelis and Yu 1997, Yu and Yang 1998). In order to obtain a minmax (regret) solution we can use a mixed integer programming formulation or a branch and bound algorithm described by Kouvelis and Yu (1997). Also, Aissi et al. (2005) showed that under the assumption that a deterministic problem is polynomially solvable, its minmax (regret) version is approximable efficiently within the number of scenarios k . However, some recent results proven by Kasperski and Zieliński (2008) suggest that the minmax (regret) versions of such basic problems as shortest path, minimum assignment and minimum cut are hard to approximate within $\log^{1-\varepsilon} k$ for any $\varepsilon > 0$. We thus can see that introducing more than one scenario significantly increases the problem complexity.

3.1 An extension of the minmax (regret) approach

In practice, a problem with scenario set Γ , such as the one shown in Example 1, may be still not appropriate. Suppose that the graph shown in Figure 1 models a part of a communication network and the cost of arc $e \in E$ is a traveling time of this arc. Two scenarios in this problem may correspond to two possible events such as traffic loads. Of course, a traffic load has a global influence on the network since an obstacle in one road influences some other roads. So, defining several different time scenarios is a good way of modeling such a situation. Notice, however, that asking about a traveling time, even assuming that a particular event will happen, we rarely get a precise answer. The traveling time is an example of a parameter whose nature is imprecise. In other words, a traveling time of a road may vary independently on the values of the traveling times of the remaining roads. Therefore, under every scenario it may be reasonable to specify a range of possible traveling times instead of a single value. We now show how such uncertainty can be taken into account in the approach described in the previous section.

Suppose first that the element costs are given as closed intervals. Hence the interval $[\underline{c}_e, \bar{c}_e]$ contains all possible values of cost of element $e \in E$. We assume that the element cost may take any value from this interval independently on the values of the costs of the remaining elements. Let $\tilde{\mathbf{c}}$ be a Cartesian product of all these intervals, namely $\tilde{\mathbf{c}} = \times_{e \in E} [\underline{c}_e, \bar{c}_e]$. Observe that $\tilde{\mathbf{c}}$ contains infinite number of possible cost realizations. Assume now that scenario set is given as $\tilde{\Gamma} = \tilde{\mathbf{c}}^1 \cup \dots \cup \tilde{\mathbf{c}}^k$, where $\tilde{\mathbf{c}}^1, \dots, \tilde{\mathbf{c}}^k$ are called *interval scenarios*. The scenario set $\tilde{\Gamma}$ generalizes scenario set Γ and models two types of uncertainty. Different interval scenarios correspond to a *structural uncertainty* having a global influence on the considered system; the intervals within every interval scenario model a *local uncertainty* connected with the imprecise nature of a single cost. We consider now the following natural generalizations of the MINMAX

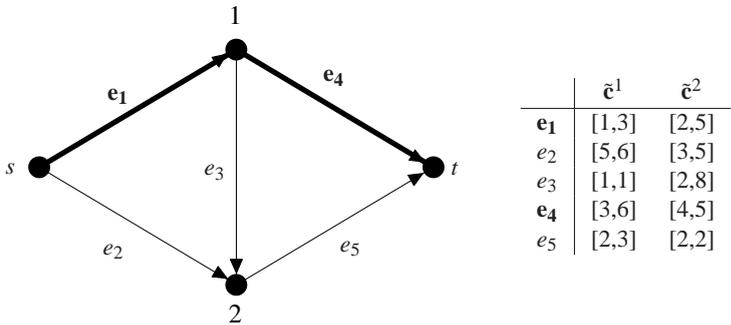
and MINMAX REGRET problems:

$$\text{MINMAX: } \min_{X \in \Phi} \max_{\mathbf{c} \in \tilde{\Gamma}} F(X, \mathbf{c}),$$

$$\text{MINMAX REGRET: } \min_{X \in \Phi} \max_{\mathbf{c} \in \tilde{\Gamma}} D(X, \mathbf{c}).$$

Example 2. Consider a shortest path problem shown in Figure 2. We have two interval scenarios $\tilde{\mathbf{c}}^1$ and $\tilde{\mathbf{c}}^2$ in the problem. So $\tilde{\mathbf{c}}^1 = [1, 3] \times [5, 6] \times \dots \times [2, 3]$ and $\tilde{\mathbf{c}}^2 = [2, 5] \times [3, 5] \times \dots \times [2, 2]$. We obtain $\tilde{\Gamma} = \tilde{\mathbf{c}}^1 \cup \tilde{\mathbf{c}}^2$. For a sample path $X = \{e_1, e_4\}$ the maximal cost over all $\mathbf{c} \in \tilde{\Gamma}$ is 10 and the maximal regret is 5. \square

Figure 2. A sample problem with two interval scenarios



Let us denote by $\bar{\mathbf{c}} \in \tilde{\mathbf{c}}$ a cost realization in which all elements $e \in E$ have costs \bar{c}_e . Obviously $\max_{\mathbf{c} \in \tilde{\mathbf{c}}} F(X, \mathbf{c}) = F(X, \bar{\mathbf{c}})$ and

$$\min_{X \in \Phi} \max_{\mathbf{c} \in \tilde{\Gamma}} F(X, \mathbf{c}) = \min_{X \in \Phi} \max_{\mathbf{c} \in \{\tilde{\mathbf{c}}^1, \dots, \tilde{\mathbf{c}}^k\}} F(X, \mathbf{c}). \tag{1}$$

Notice that (1) is a MINMAX problem with scenario set $\Gamma = \{\tilde{\mathbf{c}}^1, \dots, \tilde{\mathbf{c}}^k\}$. We thus can see that the problem with interval scenarios can be easily transformed to an equivalent MINMAX one with deterministic scenarios. We can now use any known algorithm for the MINMAX problem to solve (1). In particular, the MIP formulation and the known approximation algorithms can be directly applied. Let us denote by $\bar{\mathbf{c}}_X \in \tilde{\mathbf{c}}$ a cost realization in which all elements $e \in X$ have costs \bar{c}_e and all the remaining elements have costs c_e . It is well known (see e.g. Kasperski and Zieliński 2006) that $\max_{\mathbf{c} \in \tilde{\mathbf{c}}} D(X, \mathbf{c}) = D(X, \bar{\mathbf{c}}_X)$. In consequence

$$\min_{X \in \Phi} \max_{\mathbf{c} \in \tilde{\Gamma}} D(X, \mathbf{c}) = \min_{X \in \Phi} \max_{\mathbf{c} \in \{\bar{\mathbf{c}}_X^1, \dots, \bar{\mathbf{c}}_X^k\}} D(X, \mathbf{c}). \tag{2}$$

Contrary to (1), there is no easy transformation of (2) to the MINMAX REGRET problem with deterministic scenarios. It follows from the fact that, contrary to the MINMAX problem, the scenario set $\Gamma = \{\bar{\mathbf{c}}_X^1, \dots, \bar{\mathbf{c}}_X^k\}$ in (2) depends on solution X . The

problem (2) is not trivial even if there is only one interval scenario $\tilde{\mathbf{c}}$. In this particular case we get the following problem:

$$\min_{X \in \Phi} D(X, \bar{\mathbf{c}}_X) = \min_{X \in \Phi} \{F(X, \bar{\mathbf{c}}_X) - \hat{F}(\bar{\mathbf{c}}_X)\}. \quad (3)$$

The problem (3) is well known and widely discussed in literature (see e.g. Kasperski (2008) for a survey). This is a minmax regret discrete optimization problem with interval costs. In most cases it is NP-hard even if the deterministic problem is polynomially solvable (Aissi et al. 2005, Aron and van Hentenryck 2004, Averbakh and Lebedev 2004). However, for (3) an efficient 2-approximation algorithm is known (Kasperski and Zieliński 2006). Note that in problem (3) only the local uncertainty is taken into account.

4. Solving the minmax regret problem with a set of interval scenarios

In this section we will focus on solving the MINMAX REGRET problem with scenario set $\tilde{\Gamma}$. We will provide an exact algorithm based on a mixed integer programming formulation and we propose a simple approximation algorithm, which is fast if only the deterministic problem is polynomially solvable.

4.1 Mixed integer programming formulation

Let us assign binary variable $x_i \in \{0, 1\}$ to every element $e_i \in E$. This variable will indicate whether element e_i is contained in the constructed minmax regret solution. Every solution $X \in \Phi$ can be described by a characteristic vector $\mathbf{x} = [x_1, \dots, x_n] \in \{0, 1\}^n$ where $x_i = 1$ if and only if $e_i \in X$. We will assume that the set of all characteristic vectors can be described by some set of linear constraints of the form $\mathcal{A}\mathbf{x}^T = \mathbf{b}$, where \mathcal{A} is a matrix and \mathbf{b} is a vector of fixed coefficients. Of course, we also allow signs \leq and \geq in some constraints since they can be transformed to equalities by adding a number of additional slack variables. We will assume that matrix \mathcal{A} is totally unimodular. Recall that in a totally unimodular matrix all its nonsingular square submatrices have determinants -1 or 1. This assumption restricts the class of considered problems. However, the solutions of many important problems such as shortest path, minimum spanning tree, minimum assignment or minimum cut can be described by a system of linear constraints with a totally unimodular matrix \mathcal{A} (see e.g. Ahuja et al. 1993, Papadimitriou and Steiglitz 1998, Garfinkel and Nemhauser 1972).

To simplify notations, suppose that the j -th interval scenario $\tilde{\mathbf{c}}^j$ is Cartesian product of intervals $[\underline{c}_i^j, \bar{c}_i^j]$ for all $e_i \in E$. Using (2) we can rewrite the MINMAX REGRET problem as follows:

$$\begin{aligned} \min \lambda \\ F(X, \bar{\mathbf{c}}_X^j) - \hat{F}(\bar{\mathbf{c}}_X^j) \leq \lambda \quad j = 1, \dots, k \\ X \in \Phi \\ \lambda \geq 0, \end{aligned} \quad (4)$$

where $\hat{F}(\bar{\mathbf{c}}_X^j) = \min_{Y \in \Phi} F(Y, \bar{\mathbf{c}}_X^j)$. We fix X and consider subproblem $\min_{Y \in \Phi} F(Y, \bar{\mathbf{c}}_X^j)$. Using binary variables representing X and Y and the definition of the cost realization $\bar{\mathbf{c}}_X^j$, we can represent this problem as follows:

$$\begin{aligned} & \min \sum_{i=1}^n [\bar{c}_i^j x_i + \underline{c}_i^j (1 - x_i)] y_i \\ & \mathcal{A} \mathbf{y}^T = \mathbf{b} \\ & y_i \in \{0, 1\} \end{aligned} \quad i = 1, \dots, n$$

Using the assumption that matrix \mathcal{A} is totally unimodular, we can relax constraints $y_i \in \{0, 1\}$ obtaining the following problem that has the same minimal objective function value:

$$\begin{aligned} & \min \sum_{i=1}^n [\bar{c}_i^j x_i + \underline{c}_i^j (1 - x_i)] y_i \\ & \mathcal{A} \mathbf{y}^T = \mathbf{b} \\ & 0 \leq y_i \leq 1 \end{aligned} \quad i = 1, \dots, n \tag{5}$$

We can now construct a dual problem to (5). Let \mathbf{u}^j be a vector of dual variables, $\phi(\mathbf{u}^j)$ be the objective of the dual and let $\Psi^j(\mathbf{x})$ be the set of feasible dual vectors. So, the dual is $\max_{\mathbf{u}^j \in \Psi^j(\mathbf{x})} \phi(\mathbf{u}^j)$ and it is linear with respect to both \mathbf{u}^j and \mathbf{x} . The strong duality theorem now implies

$$\hat{F}(\bar{\mathbf{c}}_X^j) = \min_{Y \in \Phi} F(Y, \bar{\mathbf{c}}_X^j) = \max_{\mathbf{u}^j \in \Psi^j(\mathbf{x})} \phi(\mathbf{u}^j).$$

Since $F(X, \bar{\mathbf{c}}_X^j) = \sum_{i=1}^n \bar{c}_i^j x_i$, model (4) can be rewritten as follows:

$$\begin{aligned} & \min \lambda \\ & \sum_{i=1}^n \bar{c}_i^j x_i - \max_{\mathbf{u}^j \in \Psi^j(\mathbf{x})} \phi(\mathbf{u}^j) \leq \lambda \quad j = 1, \dots, k \\ & \mathcal{A} \mathbf{x}^T = \mathbf{b} \\ & x_i \in \{0, 1\} \\ & \lambda \geq 0 \end{aligned} \quad i = 1, \dots, n \tag{6}$$

We can skip the maximum operator in (6) obtaining the following equivalent model:

$$\begin{aligned} & \min \lambda \\ & \sum_{i=1}^n \bar{c}_i^j x_i - \phi(\mathbf{u}^j) \leq \lambda \quad j = 1, \dots, k \\ & \mathcal{A} \mathbf{x}^T = \mathbf{b} \\ & \mathbf{u}^j \in \Psi^j(\mathbf{x}) \\ & x_i \in \{0, 1\} \\ & \lambda \geq 0 \end{aligned} \quad j = 1, \dots, k \quad i = 1, \dots, n \tag{7}$$

The formulation (7) is a mixed integer linear programming model for MINMAX REGRET with scenario set $\tilde{\Gamma} = \tilde{\mathbf{c}}^1 \cup \dots \cup \tilde{\mathbf{c}}^k$. It can be solved by using a standard and powerful software such as CPLEX or GLPK.

Example 3. We illustrate the presented framework by an example. Suppose that $E = \{e_1, \dots, e_n\}$ is a set of items and we wish to select exactly p of them, where $p > 0$ is a given integer. This problem, called *minimum selecting items*, has been discussed by Averbakh (2001) and Conde (2004). The solution set Φ in this problem can be described by the single constraint $x_1 + x_2 + \dots + x_n = p$. Obviously, matrix $\mathcal{A} = [1, 1, \dots, 1]$ is totally unimodular. The relaxed subproblem (5) takes the following form:

$$\begin{aligned} \min \sum_{i=1}^n [\bar{c}_i^j x_i + \underline{c}_i^j (1 - x_i)] y_i \\ y_1 + y_2 + \dots + y_n = p \\ 0 \leq y_i \leq 1 \quad i = 1, \dots, n \end{aligned} \tag{8}$$

Assigning dual variable u_0^j to the equality constraint and dual variables u_1^j, \dots, u_n^j to constraints $y_i \leq 1, i = 1, \dots, n$, we get the following dual model:

$$\begin{aligned} \max pu_0^j - u_1^j - \dots - u_n^j \\ u_0^j - u_i^j \leq \bar{c}_i^j x_i + \underline{c}_i^j (1 - x_i) \quad i = 1, \dots, n \\ u_i^j \geq 0 \quad i = 1, \dots, n \end{aligned}$$

Consequently, $\phi(\mathbf{u}^j) = pu_0^j - u_1^j - \dots - u_n^j$ and set $\Psi^j(\mathbf{x})$ is described by the $2n$ constraints of the dual model. We are now ready to design the MIP model using formulation (7). This model takes the following form:

$$\begin{aligned} \min \lambda \\ \sum_{i=1}^n \bar{c}_i^j x_i - pu_0^j + \sum_{i=1}^n u_i^j \leq \lambda \quad j = 1, \dots, k \\ \sum_{i=1}^n x_i = p \\ u_0^j - u_i^j \leq \bar{c}_i^j x_i + \underline{c}_i^j (1 - x_i) \quad i = 1, \dots, n; j = 1, \dots, k \\ u_i^j \geq 0 \quad i = 1, \dots, n; j = 1, \dots, k \\ x_i \in \{0, 1\} \quad i = 1, \dots, n \\ \lambda \geq 0 \end{aligned}$$

The obtained problem can be solved by using a standard software. \square

4.2 An approximation algorithm

The main drawback of the minmax regret approach is that introducing more than one possible cost realization may dramatically increase the time required to solve the problem. Therefore, for large problems, the time required to solve the mixed integer programming model designed in the previous section may be too long. Furthermore, this model can be applied only to the problems fulfilling some specific assumptions.

In this section we design an approximation algorithm for the problem. The idea is to solve a deterministic problem for a particular cost vector constructed from $\tilde{\Gamma}$. Therefore, the approximation algorithm will be general and it can be applied to any

discrete optimization problem under the assumption that we can solve somehow its deterministic counterpart.

Before we proceed we introduce some additional notations which will simplify further considerations. Let us define $D^j(X) = D(X, \bar{c}_X^j)$ and $D(X) = \max_{j=1, \dots, k} D^j(X)$. Now, using (2), we can see the MINMAX REGRET problem is equivalent to minimizing $D(X)$ over all $X \in \Phi$. Let also denote $OPT = \min_{X \in \Phi} D(X)$, so OPT is the maximal regret of an optimal minmax regret solution. Consider ALGORITHM AM shown in Figure 3. This algorithm forms first a particular cost realization \mathbf{c} using the scenario set $\tilde{\Gamma}$ and returns then an optimal solution under \mathbf{c} .

Figure 3. An approximation algorithm for the MINMAX REGRET problem

ALGORITHM AM

Require: A MINMAX REGRET problem with scenario set $\tilde{\Gamma} = \{\tilde{\mathbf{c}}^1, \dots, \tilde{\mathbf{c}}^k\}$

Ensure: A feasible solution $Y \in \Phi$

- 1: **for all** $e \in E$ **do**
- 2: $c_e \leftarrow \sum_{j=1}^k (c_e^j + \bar{c}_e^j)$ {Form a cost vector \mathbf{c} }
- 3: **end for**
- 4: Output an optimal solution $Y \in \Phi$ under cost vector \mathbf{c}

Notice that for $k = 1$ ALGORITHM AM boils down to the 2-approximation algorithm constructed by Kasperski and Zieliński (2006). ALGORITHM AM can also be viewed as a generalization of the k -approximation algorithm proposed by Aissi et al. (2007). If we apply the algorithm to the problem from Example 2, then we need to solve a deterministic shortest path problem for the cost vector $[11, 19, 12, 18, 9]$ and as a result we get path $X = \{e_2, e_5\}$. We now explore the quality of a solution returned by ALGORITHM AM.

Theorem 1. ALGORITHM AM outputs a solution $Y \in \Phi$ such that $D(Y) \leq 2k * OPT$.

Proof. Let $X \in \Phi$ and $Y \in \Phi$ be two feasible solutions. The following two formulas have been established by Kasperski and Zieliński (2006):

$$D^j(X) \geq \sum_{e \in X \setminus Y} \bar{c}_e^j - \sum_{e \in Y \setminus X} c_e^j \tag{9}$$

$$D^j(Y) \leq D^j(X) + \sum_{e \in Y \setminus X} \bar{c}_e^j - \sum_{e \in X \setminus Y} c_e^j \tag{10}$$

Inequalities (9) and (10) imply

$$\sum_{j=1}^k D^j(X) \geq \sum_{j=1}^k \left[\sum_{e \in X \setminus Y} \bar{c}_e^j - \sum_{e \in Y \setminus X} c_e^j \right], \tag{11}$$

$$\sum_{j=1}^k D^j(Y) \leq \sum_{j=1}^k D^j(X) + \sum_{j=1}^k \left[\sum_{e \in Y \setminus X} \bar{c}_e^j - \sum_{e \in X \setminus Y} c_e^j \right]. \tag{12}$$

Assume that Y is the solution returned by ALGORITHM AM. Then, for any $X \in \Phi$

$$\sum_{e \in Y} \sum_{j=1}^k (\underline{c}_e^j + \bar{c}_e^j) \leq \sum_{e \in X} \sum_{j=1}^k (\underline{c}_e^j + \bar{c}_e^j),$$

which, after simple algebraic manipulations, is equivalent to

$$\sum_{j=1}^k \left[\sum_{e \in X \setminus Y} \bar{c}_e^j - \sum_{e \in Y \setminus X} \underline{c}_e^j \right] \geq \sum_{j=1}^k \left[\sum_{e \in Y \setminus X} \bar{c}_e^j - \sum_{e \in X \setminus Y} \underline{c}_e^j \right]. \quad (13)$$

Now formulas (11) and (13) imply

$$\sum_{j=1}^k D^j(X) \geq \sum_{j=1}^k \left[\sum_{e \in Y \setminus X} \bar{c}_e^j - \sum_{e \in X \setminus Y} \underline{c}_e^j \right],$$

which together with (12) yield

$$\sum_{j=1}^k D^j(Y) \leq 2 \sum_{j=1}^k D^j(X). \quad (14)$$

Inequality (14) implies

$$\max_{j=1, \dots, k} D^j(Y) \leq 2k * \max_{j=1, \dots, k} D^j(X).$$

In particular, if $D(X) = \max_{j=1, \dots, k} D^j(X) = OPT$, then $D(Y) = \max_{j=1, \dots, k} D^j(Y) \leq 2k * OPT$, which completes the proof. \square

Observe that if the deterministic problem is polynomially solvable, then ALGORITHM AM runs in polynomial time. So, it can be used to obtain approximate solutions for very large problems. However, this algorithm can also be applied if the deterministic problem is NP-hard. But in this case its running time may be not polynomial.

4.3 Computational tests

In this section we present some results of computational tests. Our aim is to compare the MIP formulation to ALGORITHM AM designed in the previous section. We wish to identify the factors that have the most influence on the computation times and on the quality of solutions returned by ALGORITHM AM. The tests were performed on the minimum selecting items problem described in Section 4.1 (see Example 3). Let us denote by (n, k, d) a family of minimum selecting items problems where:

- (i) n is the number of items. We fix $p = \lceil n/2 \rceil$ to obtain the largest solution space.

Notice that the size of Φ is $\binom{n}{\lceil n/2 \rceil}$, which becomes a huge number for rather small n .

- (ii) k is the number of interval scenarios.

- (iii) d is a degree of uncertainty of the interval scenarios. Namely, every interval $[c_i^j, \bar{c}_i^j]$ is generated randomly so that it is fully contained in the interval $[0,100]$ and its width is such that $\bar{c}_i^j - c_i^j \leq d$.

In our tests we have chosen $n = 60, 80, 100$, $k = 2, 3, 4, 5$ and $d = 20, 40, 60$. For every combination of n , k and d we have generated and solved 10 instances. All the tests were performed on a computer equipped with a Core Duo 1.8GHz processor with 1GB RAM. The GLPK 4.21 solver was used to solve the MIP models. The obtained results are shown in Table 1. In column tm. the average computation times in seconds required to solve the MIP formulation and obtain an optimal solution are shown. In the next three columns the percentage average, minimum and maximum deviations from optimum, i.e. the values of $100(D(Y) - OPT)/OPT$, reported for solutions Y returned by ALGORITHM AM are shown.

Table 1. The results of computational tests

family	tm.	av.	min	max	family	tm.	av.	min	max	family	tm.	av.	min	max
(60,2,20)	0.0	17.0	0.0	29.8	(80,2,20)	0.0	9.7	0.0	30.1	(100,2,20)	0.0	12.0	2.1	21.7
(60,2,40)	0.0	18.8	5.3	40.6	(80,2,40)	0.0	14.8	7.8	19.9	(100,2,40)	0.6	17.8	2.5	32.1
(60,2,60)	0.4	16.9	4.16	27.0	(80,2,60)	0.9	12.0	5.6	20.7	(100,2,60)	6.1	8.5	0.1	24.6
(60,3,20)	0.4	17.8	0.0	40.4	(80,3,20)	0.9	13.1	7.4	27.8	(100,3,20)	1.8	16.4	3.0	31.1
(60,3,40)	1.0	19.9	6.0	40.9	(80,3,40)	1.5	14.8	2.9	30.0	(100,3,40)	5.6	16.6	5.7	23.0
(60,3,60)	6.8	19.1	3.2	32.1	(80,3,60)	16.4	14.0	6.2	20.9	(100,3,60)	369.4	17.3	8.1	26.0
(60,4,20)	1.8	19.0	6.2	34.1	(80,4,20)	4.6	22.6	9.0	48.5	(100,4,20)	9.1	16.4	5.5	30.2
(60,4,40)	3.5	19.0	2.7	31	(80,4,40)	10.5	14.6	2.0	31.0	(100,4,40)	44.5	14.9	6.2	33.4
(60,4,60)	57.4	14.8	4.3	27.2	(80,4,60)	421.2	16.0	3.4	33.3	(100,4,60)	1074.2	16.0	6.8	30.0
(60,5,20)	5.5	21.6	6.5	56.1	(80,5,20)	17.0	23.2	4.6	49.0	(100,5,20)	65.6	16.9	7.2	22.8
(60,5,40)	6.1	20.1	13.5	25.6	(80,5,40)	28.2	22.7	9.0	30.3	(100,5,40)	254.0	16.5	11.0	25.0
(60,5,60)	42.8	19.0	9.6	37.0	(80,5,60)	558.5	17.1	7.3	25.6	(100,5,60)	>3600	?	?	?

Note: ‘tm.’ denotes the average computation times in seconds required to solve the MIP formulation; ‘av.’, ‘min’, ‘max’ denote the percentage average, minimum and maximum deviations from optimum reported for a solution returned by ALGORITHM AM, respectively.

As we can see from the obtained results, the computation times increase with the number of items and the number of scenarios, which is not surprising. However, the increase with the number of scenarios is very quick - especially for the families with 100 items. The MIP formulation is efficient only if the number of scenarios is a small number. Interestingly, the computation times also increase with the degree of uncertainty d . Notice that we were unable to solve the family (100, 5, 60) within less than 1 hour. For larger problems, having a large number of interval scenarios, ALGORITHM AM should be used. Observe that the largest percentage deviation from optimum reported for all generated 350 instances is 56.1%, while the average percentage deviation over all instances is about 17%. So, the average performance of the approximation algorithm seems to be much better than its theoretical worst case behaviour. Therefore,

the simple approximation algorithm is a good choice if the MIP solver fails to compute an optimal solution in a reasonable time.

We have performed the tests for a particular problem. We conjecture, however, that a similar performance will be reported for other problems. In particular, the same factors will influence on the computation times.

5. Conclusions

In this paper we have proposed an extension of the known minmax (regret) approach to discrete optimization. This approach allows us to model two different types of uncertainty. The first, called a structural uncertainty, models some unpredictable events having a global effect on a considered system and the second, called a local uncertainty, is connected with an imprecise nature of costs. We have introduced scenario set $\tilde{\Gamma}$ being an union of a number of interval scenarios. In order to choose a solution we have applied minmax and minmax regret criteria. In most cases the discussed approach leads to problems which are computationally hard. Having a particular problem one can try to apply a mixed integer programming formulation to obtain a solution. If it is not possible or an optimal solution cannot be obtained in a reasonable time, then the proposed approximation algorithm can be used.

References

- Ahuja, R. K., Magnanti, T. L. and Orlin, J. B. (1993). *Network Flows, Theory, Algorithms and Applications*. New Jersey, Prentice Hall.
- Aissi, H., Bazgan, C. and Vanderpooten, D. (2005). Complexity of the Min-max (Regret) Versions of Min Cut Problems. *Discrete Optimization*, 5(1), 66–73.
- Aissi, H., Bazgan, C. and Vanderpooten, D. (2007). Approximation of Min-max and Min-max Regret Versions of Some Combinatorial Optimization Problems. *European Journal of Operational Research*, 179(2), 281–290.
- Aissi, H., Bazgan, C. and Vanderpooten, D. (2008). Minmax and Minmax Regret Versions of Combinatorial Optimization Problems: a Survey. (forthcoming in *European Journal of Operational Research*, DOI: 10.1016/j.ejor.2008.09.012)
- Aron, I. and van Hentenryck, P. (2004). On the Complexity of the Robust Spanning Tree Problem with Interval Data. *Operations Research Letters*, 32(1), 36–40.
- Averbakh, I. (2001). On the Complexity of a Class of Combinatorial Optimization Problems with Uncertainty. *Mathematical Programming*, A 90, 263–272.
- Averbakh, I. and Lebedev, V. (2004). Interval Data Minmax Regret Network Optimization Problems. *Discrete Applied Mathematics*, 138(3), 289–301.
- Conde, E. (2004). An Improved Algorithm for Selecting p Items with Uncertain Returns according to the Minmax-regret Criterion. *Mathematical Programming*, A 100, 345–353.

Garfinkel, R. S. and Nemhauser, G. L. (1972). *Integer Programming*. New York, John Wiley and Sons.

Kasperski, A. and Zieliński, P. (2006). An Approximation Algorithm for Interval Data Minmax Regret Combinatorial Optimization Problems. *Information Processing Letters*, 97(5), 177–180.

Kasperski, A. and Zieliński, P. (2008). On the Approximability of Minmax (Regret) Network Optimization Problems. (forthcoming in *Information Processing Letters*, DOI: 10.1016/j.ipl.2008.10.008)

Kasperski, A. (2008). Discrete Optimization with Interval Data. Minmax Regret and Fuzzy Approach. *Studies in Fuzziness and Soft Computing*, vol. 228. Berlin, Heidelberg, Springer-Verlag.

Kouvelis, P. and Yu, G. (1997). *Robust Discrete Optimization and Its Applications*. Boston, Kluwer Academic Publishers.

Luce, R. D. and Raiffa, H. (1957). *Games and Decisions. Introduction and Critical Survey*. New York, Dover Publications, Inc.

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(3), 225–232.

Montemanni, R. and Gambardella, L. M. (2005). A Branch and Bound Algorithm for the Robust Spanning Tree Problem with Interval Data. *European Journal of Operational Research*, 161(3), 771–779.

Papadimitriou, C. and Steiglitz, K. (1998). *Combinatorial Optimization. Algorithms and Complexity*. Mineola, New York, Dover Publications, Inc.

Savage, L. J. (1951). The Theory of Statistical Decisions. *Journal of the American Statistical Association*, 46, 55–67.

Yaman, H., Karasan, O. E. and Pinar, M. C. (2001). The Robust Spanning Tree Problem with Interval Data. *Operations Research Letters*, 29(1), 31–40.

Yu, G. and Yang, J. (1998). On the Robust Shortest Path Problem. *Computers and Operations Research*, 25(6), 457–468.