Pseudo-Polynomial Algorithms for Min-Max and Min-Max Regret Problems[*]

Hassene Aissi   Cristina Bazgan   Daniel Vanderpooten

LAMSADE, Université Paris-Dauphine, France
Email: {aissi,bazgan,vdp}@lamsade.dauphine.fr

Abstract *We present in this paper general pseudo-polynomial time algorithms to solve min-max and min-max regret versions of some polynomial or pseudo-polynomial problems under a constant number of scenarios. Using easily computable bounds, we can improve these algorithms. This way we provide pseudo-polynomial algorithms for the min-max and and min-max regret versions of several classical problems including minimum spanning tree, shortest path, and knapsack.*

Keywords min-max, min-max regret, computational complexity, pseudo-polynomial

1 Introduction

The definition of an instance of a combinatorial optimization problem requires to specify parameters, in particular objective function coefficients, which may be uncertain or imprecise. Uncertainty/imprecision can be structured through the concept of *scenario* which corresponds to an assignment of plausible values to model parameters. Each scenario s can be represented as a vector in $\mathbb{R}^m$ where m is the number of relevant numerical parameters. Kouvelis and Yu [2] proposed the maximum cost and maximum regret criteria, stemming from decision theory, to construct solutions hedging against parameters variations. In min-max optimization, the aim is to find a solution having the best worst case value across all scenarios. In min-max regret problem, it is required to find a feasible solution minimizing the maximum deviation, over all possible scenarios, of the value of the solution from the optimal value of the corresponding scenario. Two natural ways of describing the set of all possible scenarios S have been considered in the literature. In the *interval data case*, each numerical parameter can take any value between a lower and upper bound, independently of the values of the other parameters. Thus, in this case, S is the cartesian product of the intervals of uncertainty for the parameters. In the *discrete scenario case*, S is described explicitly by the list of

*This work has been partially funded by grant CNRS/CGRI-FNRS number 18227. The second author was partially supported by the ACI Scurit Informatique grant-TADORNE project 2004.

all vectors $s \in S$. In this case, that is considered in this paper, we distinguish situations where the number of scenarios is bounded by a constant from those where the number of scenarios is unbounded.

Complexity of the min-max (regret) versions has been studied extensively during the last decade. In [2], the complexity of min-max (regret) versions of several combinatorial optimization problems was studied, including shortest path and minimum spanning tree. In general, these versions are shown to be harder than the classical versions.

More precisely, if the number of scenarios is not constant, these problems become strongly *NP*-hard, even when the classical problems are solvable in polynomial time.

On the other hand, for a constant number of scenarios, it was only partially known if these problems are strongly or weakly *NP*-hard. Indeed, the reductions described in [2] to prove *NP*-difficulty are based on transformations from the partition problem which is known to be weakly *NP*-hard. These reductions give no indication as to the precise status of these problems. The only known weakly *NP*-hard problems are those for which there exists a pseudo-polynomial algorithm based on dynamic programming (shortest path, knapsack, minimum spanning tree on grid graphs, ...). We present in this paper general pseudo-polynomial algorithms for the min-max (regret) versions of problems whose exact version and the associated search version are pseudo-polynomial. These algorithms enable us to obtain alternative pseudo-polynomial algorithms for min-max regret (versions) of shortest path and knapsack. Furthermore, we obtain pseudo-polynomials algorithms for min-max regret (versions) of spanning tree in general graphs and weighted perfect matching in planar graphs.

After introducing some preliminaries in Section 2, we develop general pseudo-polynomial algorithms for min-max and min-max regret problems in Section 3. Improved versions of these algorithms are obtained in Section 4 using bounds that can be easily determined.

2 Preliminaries

We consider in this paper the class of problems with a linear objective function defined as:

$$\begin{cases} \min \sum_{i=1}^{m} c_i x_i & c_i \in \mathbb{N} \\ x \in X \subset \{0,1\}^m \end{cases}$$

This class encompasses a large variety of classical combinatorial problems, some of which are polynomial-time solvable (shortest path problem, minimum spanning tree, ...) and others are *NP*-difficult (knapsack, set covering, ...).

In this paper, we focus on the subclass $\mathcal{C}$ of problems which can be solved in polynomial or pseudo-polynomial time.

Denote by $t_o(|I|)$ the running time of a (pseudo-)polynomial time algorithm solving instance I of a problem $\mathcal{P} \in \mathcal{C}$.

Given a problem $\mathcal{P} \in \mathcal{C}$, the min-max (regret) version associated to $\mathcal{P}$ has for input a finite set of scenarios S where each scenario $s \in S$ is represented by a vector $(c_1^s, \ldots, c_m^s)$. We denote by $val(x, s) = \sum_{i=1}^{m} c_i^s x_i$ the value of solution $x \in X$ under scenario $s \in S$ and by val_s^* the optimal value in scenario s.

The min-max optimization problem corresponding to $\mathcal{P}$, denoted by Min-Max $\mathcal{P}$, consists of finding a solution x having the best worst case value across all scenarios, which can be stated as:

$$\min_{x \in X} \max_{s \in S} val(x, s)$$

Given a solution $x \in X$, its *regret*, $R(x, s)$, under scenario $s \in S$ is defined as $R(x, s) = val(x, s) - val_s^*$. The *maximum regret* $R_{max}(x)$ of solution x is then defined as $R_{max}(x) = \max_{s \in S} R(x, s)$.

The min-max regret optimization problem corresponding to $\mathcal{P}$, denoted by Min-Max Regret $\mathcal{P}$, consists of finding a solution x minimizing the maximum regret $R_{max}(x)$ which can be stated as:

$$\min_{x \in X} R_{max}(x) = \min_{x \in X} \max_{s \in S} \{val(x, s) - val_s^*\}$$

When $\mathcal{P}$ is a maximization problem, the max-min and min-max regret versions associated to $\mathcal{P}$ are defined similarly.

In order to obtain our algorithms, we need to solve the *exact* version of a problem $\mathcal{P}$, denoted by Exact $\mathcal{P}$, which consists of deciding whether there exists a solution with value exactly v, for some specified v. The associated search problem consists of finding such a solution whenever it exists. This notion was first discussed in [4]. In [1] a pseudo-polynomial algorithm is given for the exact spanning tree problem and for the exact perfect matching problem in planar graphs.

The procedure for deciding if there is a solution of value exactly *value* in an instance I of Exact $\mathcal{P}$ is called Decide_Exact_$\mathcal{P}$(I,*value*) and its running time is denoted by $t_d(|I|)$. The procedure used to construct a solution x with value exactly *value* for an instance I of $\mathcal{P}$ is called Construct_Exact_$\mathcal{P}$(I,*value*,x) and its running time is denoted by $t_c(|I|)$.

3 General pseudo-polynomial algorithms

In this section, we present general pseudo-polynomial algorithms to solve the min-max (regret) versions of some polynomial or pseudo-polynomial time solvable problems.

Theorem 1. *There is a pseudo-polynomial time algorithm for* Min-Max (Regret) $\mathcal{P}$ *if* Exact $\mathcal{P}$ *and its search version are polynomially or pseudo-polynomially solvable.*

Proof. Let $\mathcal{P}$ be a minimization problem such that Exact $\mathcal{P}$ and its search version are polynomially or pseudo-polynomially solvable. Consider I an instance of Min-Max $\mathcal{P}$ defined on a set S of k scenarios $s_1, \ldots, s_k$, where each scenario $s \in S$ is represented by $(c_1^s, \ldots, c_m^s)$. Let $M = \max_{i,s} c_i^s$.

The procedure for solving I consists of scanning in increasing order over all possible values $v = 0, \ldots, mM$ and deciding whether there exists a feasible solution x of I with value α_p in the pth scenario for $p = 1, \ldots, k$ and $v = \max\{\alpha_1, \ldots, \alpha_k\}$. The k conditions can be equivalently transformed into a unique condition

$$\sum_{p=1}^{k} val(x,p)(mM+1)^{p-1} = \sum_{p=1}^{k} \alpha_p (mM+1)^{p-1}.$$

Hence, this decision problem is equivalent to deciding whether there exists a solution with value exactly $\sum_{p=1}^{k} \alpha_p (mM+1)^{p-1}$ in an instance I' of Exact $\mathcal{P}$ deduced from I by aggregating the evaluations in the different scenarios. The coefficients of the objective function in I' are $c'_i = \sum_{p=1}^{k} c_i^p (mM+1)^{p-1}$ for $i = 1, \ldots, m$. Algorithm 1 computes an optimal solution for the min-max version of $\mathcal{P}$, as illustrated in Figure 1.

Algorithm 1 Pseudo-polynomial algorithm for min-max version

```
$v \leftarrow 0$
$test \leftarrow false$
while $test \neq true$ do
   for $(\alpha_1, \ldots, \alpha_k)$: $\max\{\alpha_1, \ldots, \alpha_k\} = v$ do
      $value \leftarrow \sum_{p=1}^{k} \alpha_p (mM+1)^{p-1}$
      if DECIDE_EXACT_$\mathcal{P}$($I'$,$value$) then
         CONSTRUCT_EXACT_$\mathcal{P}$ ($I'$,$value$,$x^*$)
         $test \leftarrow true$
   if $test \neq true$ then
      $v \leftarrow v + 1$
$opt \leftarrow v$
Output $opt$ and $x^*$
```

The running time of Algorithm 1 is $O(m^k M^k t_d(|I'|) + t_c(|I'|))$. The maximum value appearing in I' is $O(\frac{(mM)^k}{m})$. Since Exact $\mathcal{P}$ and its search version are solvable in polynomial or pseudo-polynomial time, there exists a constant c such that $t_d(|I'|), t_c(|I'|) \leq (m^{k-1} M^k)^c$. Thus Algorithm 1 is pseudo-polynomial.

Consider now an instance I of Min-Max Regret $\mathcal{P}$. We first compute val_s^* for all $s \in S$. The procedure now consists of scanning in increasing order all possible values $v = 0, \ldots, mM - \max_{s \in S} val_s^*$ and deciding whether there exists a feasible solution x with $val(x,p) = \alpha_p$ for $p = 1, \ldots, k$ where $\alpha_p \leq val_p^* + v$ and there exists $q \leq k$ with $\alpha_q = val_q^* + v$. Algorithm 2 computes an optimal solution for the min-max regret version of $\mathcal{P}$, as illustrated in Figure 2. The running time of Algorithm 2 is $O(kt_o(|I|/k) + m^k M^k t_d(|I'|) + t_c(|I'|))$. Clearly this algorithm is pseudo-polynomial.

For a maximization problem, max-min and min-max regret can be solved in a similar way. □

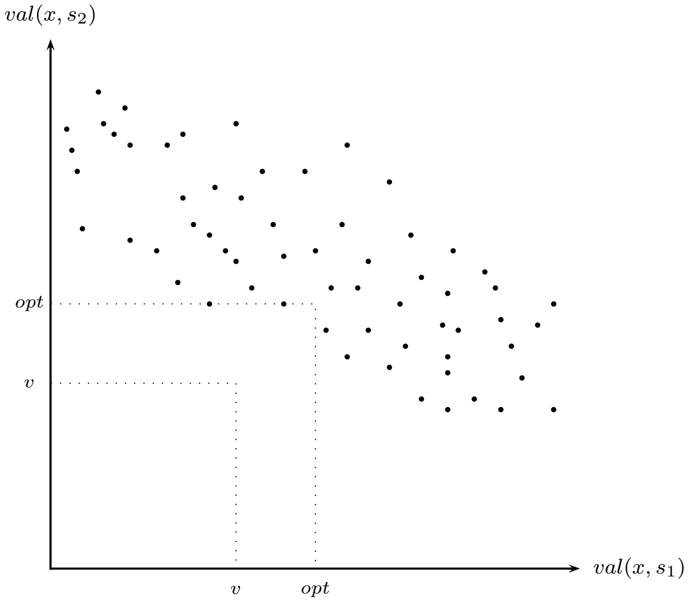

Figure 1: Illustration of Algorithm 1 for $k = 2$

Corollary 1. Min-Max (Regret) Spanning Tree, Min-Max (Regret) Shortest Path, Min-Max (Regret) Weighted Perfect Matching *in planar graphs,* Max-Min Knapsack, Min-Max Regret Knapsack, *are solvable in pseudo-polynomial time.*

Proof. Pseudo-polynomial algorithms for the exact versions of the Spanning Tree problem and Weighted Perfect Matching in planar graphs are described in [1]. The search versions of these problems can be solved by using self reducibility [3]. Exact versions of Shortest Path and Knapsack can be solved in pseudo-polynomial time using dynamic programming. □

4 Improved algorithms using bounds

In order to obtain more efficient pseudo-polynomial time algorithms for the min-max and min-max regret versions, we may use an upper and a lower bound of the optimum value. Such bounds can be easily determined as shown by the following result.

Theorem 2. *For any instance on a set of k scenarios of* Min-Max $\mathcal{P}$ *and* Min-Max Regret $\mathcal{P}$*, there exist a lower and an upper bound L and U of opt, such that $U \leq kL$. Moreover, if $\mathcal{P}$ is solvable in (pseudo-)polynomial time, then L and U are computable in (pseudo-)polynomial time.*

Algorithm 2 Pseudo-polynomial algorithm for min-max regret version

```
Compute $val^*_s$ for all $s \in S$
$v \leftarrow 0$
$test \leftarrow false$
while $test \neq true$ do
  for $(\alpha_1, \ldots, \alpha_k) : max\{\alpha_1 - val^*_1, \ldots, \alpha_k - val^*_k\} = v$ do
    $value \leftarrow \sum_{p=1}^{k} \alpha_p (mM+1)^{p-1}$
    if DECIDE_EXACT_$\mathcal{P}(I', value)$ then
      CONSTRUCT_EXACT_$\mathcal{P}(I', value, x^*)$
      $test \leftarrow true$
  if $test \neq true$ then
    $v \leftarrow v + 1$
$opt \leftarrow v$
Output $opt$ and $x^*$
```

Proof. Consider an instance I of MIN-MAX $\mathcal{P}$ defined on a set S of k scenarios where each scenario $s \in S$ is represented by $(c^s_1, \ldots, c^s_m)$ and let X be the set of feasible solutions of I. We define the following instance I' of a single scenario problem $\min_{x \in X} \sum_{s \in S} \frac{1}{k} val(x,s)$ obtained by taking objective function coefficients $c'_i = \sum_{s=1}^{k} \frac{c^s_i}{k}$, $i = 1, \ldots, m$. Let $\widetilde{x}$ be an optimal solution of I'. We take as lower and upper bounds $L = \sum_{s \in S} \frac{1}{k} val(\widetilde{x}, s)$ and $U = \max_{s \in S} val(\widetilde{x}, s)$. Clearly, we have

$$L = \min_{x \in X} \sum_{s \in S} \frac{1}{k} val(x,s) \leq \min_{x \in X} \sum_{s \in S} \frac{1}{k} (\max_{s \in S} val(x,s)) = \min_{x \in X} \max_{s \in S} val(x,s) = opt$$

and

$$\min_{x \in X} \max_{s \in S} val(x,s) \leq \max_{s \in S} val(\widetilde{x}, s) \leq \sum_{s \in S} val(\widetilde{x}, s) = k \sum_{s \in S} \frac{1}{k} val(\widetilde{x}, s) = kL$$

Consider now an instance I of MIN-MAX REGRET $\mathcal{P}$ defined on a set S of k scenarios and let X be the set of feasible solutions of I. Let $\widetilde{x} \in X$ be an optimal solution of the single scenario instance I' derived from I as for the min-max case. We take as lower and upper bounds $L = \sum_{s \in S} \frac{1}{k}(val(\widetilde{x}, s) - val^*_s)$ and $U = \max_{s \in S}(val(\widetilde{x}, s) - val^*_s)$. Clearly, we have

$$L = \min_{x \in X} \frac{1}{k} \sum_{s \in S} (val(x,s) - val^*_s) \leq \min_{x \in X} \frac{1}{k} k \max_{s \in S} (val(x,s) - val^*_s) = opt$$

and

$$\min_{x \in X} \max_{s \in S} (val(x,s) - val^*_s) \leq \max_{s \in S} (val(\widetilde{x}, s) - val^*_s) \leq \sum_{s \in S} (val(\widetilde{x}, s) - val^*_s) = kL$$

If any instance of $\mathcal{P}$ of size n is solvable in time $p(n)$, where p is a polynomial, then L and U are computable in $p(|I|/k)$. Considering $M = \max_{i,s} c^s_i$, if any

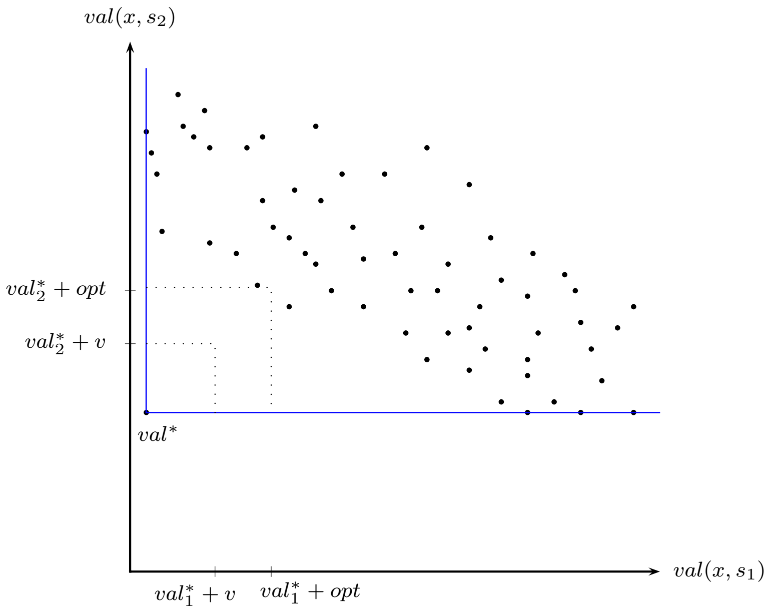

Figure 2: Illustration of Algorithm 2 for $k = 2$

instance of $\mathcal{P}$ of size n is solvable in time $p(n, M)$, where p is a polynomial, then L and U are computable in $p(|I|/k, M)$. □

The search space can be narrowed by using bounds L and U. For solving min-max versions, only solutions $x \in X$ verifying $L \leq \max_{s \in S} val(x, s) \leq U$ and $\sum_{p=1}^{k} val(x, p) \geq kL$ are to be checked. This gives rise to Algorithm 3, illustrated in Figure 3, whose running time is now $O(t_o(|I|/k) + (U-1)^k t_d(|I'|) + t_c(|I'|))$.

Similar modifications and arguments hold for the min-max regret versions, giving an improved algorithm with running time $O((k+1)t_o(|I|/k) + (U-1)^k t_d(|I'|) + t_c(|I'|))$.

References

[1] F. Barahona and R. Pulleyblank. Exact arborescences, matching and cycles. *Discrete Applied Mathematics*, 16:91–99, 1987.

[2] P. Kouvelis and G. Yu. *Robust Discrete Optimization and its Applications.* Kluwer Academic Publishers, Boston, 1997.

[3] C. H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.

[4] C. H. Papadimitriou and M. Yannakakis. The complexity of restricted spanning tree. *Journal of the Association for Computing Machinery*, 29(2):285–309, April 1982.

Algorithm 3 Improved pseudo-polynomial algorithm for min-max version

```
Compute $L$, $U$ and $\widetilde{x}$
$v \leftarrow L$
$test \leftarrow false$
while $test \neq true$ and $v \leq U - 1$ do
  for $(\alpha_1, \ldots, \alpha_k)$: $\max\{\alpha_1, \ldots, \alpha_k\} = v$ and $\sum_{p=1}^{k} \alpha_p \geq kL$ do
    $value \leftarrow \sum_{p=1}^{k} \alpha_p (mM+1)^{p-1}$
    if DECIDE_EXACT_$\mathcal{P}$($I'$,$value$) then
      CONSTRUCT_EXACT_$\mathcal{P}$ ($I'$,$value$,$x^*$)
      $test \leftarrow true$
  if $test \neq true$ then
    $v \leftarrow v + 1$
if $test = true$ then
  $opt \leftarrow v$
else
  $opt \leftarrow U$ and $x^* \leftarrow \widetilde{x}$
Output $opt$ and $x^*$
```

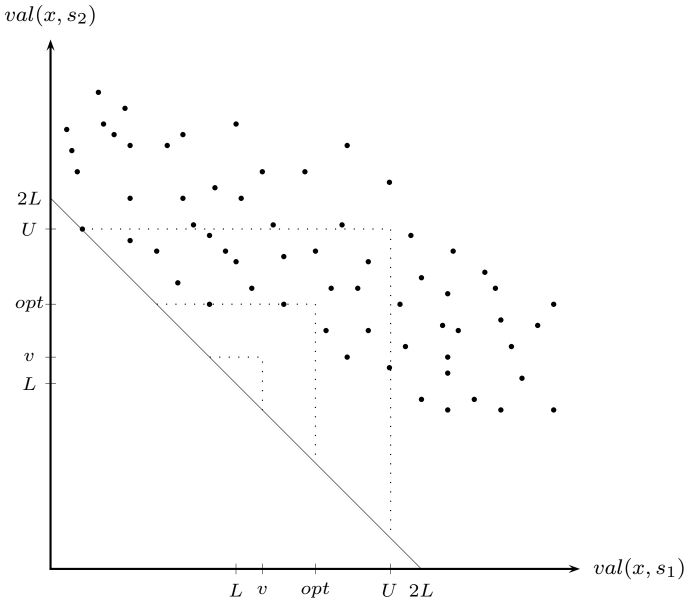

Figure 3: Illustration of Algorithm 3 for $k = 2$