# General approximation schemes for min-max (regret) versions of some (pseudo-)polynomial problems*

Hassene Aissi Cristina Bazgan Daniel Vanderpooten

Université Paris-Dauphine, LAMSADE
Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France
{aissi,bazgan,vdp}@lamsade.dauphine.fr

**Abstract**

While the complexity of min-max and min-max regret versions of most classical combinatorial optimization problems has been thoroughly investigated, there are very few studies about their approximation. For a bounded number of scenarios, we establish general approximation schemes which can be used for min-max and min-max regret versions of some polynomial or pseudo-polynomial problems. Applying these schemes to shortest path, minimum spanning tree, minimum weighted perfect matching on planar graphs, and knapsack problems, we obtain fully polynomial-time approximation schemes with better running times than the ones previously presented in the literature.

**Keywords:** min-max, min-max regret, approximation, fptas, shortest path, minimum spanning tree, knapsack, minimum weighted perfect matching.

## 1 Introduction

The definition of an instance of a combinatorial optimization problem requires one to specify parameters, in particular objective function coefficients, which may be uncertain or imprecise. Uncertainty/imprecision can be structured through the concept of a *scenario* which corresponds to an assignment of plausible values to parameters. There exist two natural ways of describing the set of all possible scenarios. In the *interval data case*, each numerical parameter can take any value between a lower bound and an upper bound. In the *discrete scenario case*, which is considered here, the scenario set is described explicitly. Kouvelis and Yu [11] proposed the min-max and min-max regret criteria, stemming from decision theory, to construct solutions hedging against parameters variations. The min-max criterion aims at constructing solutions having the best performance in the worst case. The min-max regret criterion, less conservative, aims at obtaining a solution minimizing, over all possible scenarios, the maximum deviation between the value of the solution and the optimal value for the corresponding scenario. A recent survey about complexity, approximation, and exact resolution of min-max and min-max regret versions of classical combinatorial optimization problems can be found in [3].

*This work is partially supported by the ANR project GUEPARD

The complexity of the min-max and min-max regret versions has been studied extensively during the last decade. In [11], for the discrete scenario case, 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, min-max (regret) versions of these polynomial problems usually become weakly *NP*-hard.

While the complexity of these problems was studied thoroughly, their approximation was not studied until now, except in [2]. That paper investigated the relationships between min-max (regret) and multi-objective versions, and showed the existence, in the case of a constant number of scenarios, of fully polynomial-time approximation schemes (fptas's) for min-max versions of several classical optimization problems (shortest path, minimum spanning tree, knapsack). The interest of studying these relationships is that, unlike for min-max (regret) versions, fptas's which determine an approximation of the non-dominated set (or Pareto set) have been proposed for the multi-objective version (see, e.g., [15, 18]). Approximation algorithms for the min-max version, which basically consist of selecting one min-max solution from an approximation of the non-dominated set, are then easy to derive but critically depend on the running time of the approximation scheme for the multi-objective version.

In this paper, we adopt an alternative perspective and develop general approximation schemes in the case of a constant number of scenarios, based on the scaling technique, which can be applied to the min-max/max-min and min-max regret versions of some problems, provided that some general conditions are satisfied. The advantage of this approach is that the resulting fptas's usually have a much better running time than those derived using multi-objective fptas's.

After presenting some background concepts in Section 2, we introduce the general approximation schemes in Section 3. In Section 4, we present applications of these general schemes to shortest path, minimum spanning tree, minimum weighted perfect matching in planar graphs, and knapsack problems, giving in each case fptas's with better running times than previously known fptas's based on multi-objective versions.

## 2 Preliminaries

We consider in this paper the class $\mathcal{C}$ of 0-1 problems with a linear objective function defined as:

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

This class encompasses a large variety of classical combinatorial problems, some of which are polynomial-time solvable (shortest path, minimum spanning tree, ...) and others are *NP*-hard (knapsack, set covering, ...). The *size* of a solution $x \in X$ is the number of variables $x_i$ which are set to 1.

### 2.1 Min-max, max-min and min-max regret versions

Given a problem $\mathcal{P} \in \mathcal{C}$, the min-max, max-min, and min-max regret versions associated to $\mathcal{P}$ have for input a finite set of scenarios $S$, $|S| \geq 2$, 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 or max-min version associated to a problem $\mathcal{P}$ consists of finding a solution having the best worst-case value across all scenarios. More precisely, for a minimization problem $\mathcal{P}$, its min-max version, denoted by MIN-MAX $\mathcal{P}$, can be stated as $\min_{x\in X} \max_{s\in S} val(x,s)$. For a maximization problem $\mathcal{P}$, its max-min version, denoted by MAX-MIN $\mathcal{P}$, can be stated as $\max_{x\in X} \min_{s\in S} val(x,s)$

Given a solution $x \in X$, its *regret* under scenario $s \in S$ is defined as $R(x,s) = val(x,s) - val_s^*$ for minimization problems and $R(x,s) = val_s^* - val(x,s)$ for maximization problems. 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

$$\min_{x\in X} R_{max}(x)$$

### 2.2 Approximation

Let us consider an instance $I$, of size $|I|$, of an optimization problem and a solution $x$ of $I$. We denote by $c_{max}(I)$ the maximal value of the coefficients in the objective function and $opt(I)$ the optimum value of instance $I$. The *performance ratio* of $x$ is $r(x) = \max\left\{\frac{val(x)}{opt(I)}, \frac{opt(I)}{val(x)}\right\}$.

For a function $f$, an algorithm is an $f(n)$-*approximation algorithm* if, for any instance $I$ of size $n$ of the problem, it returns a solution $x$ such that $r(x) \leq f(n)$. An optimization problem has a *fully polynomial-time approximation scheme* (an *fptas*, for short) if, for every constant $\varepsilon > 0$, it admits an $(1+\varepsilon)$-approximation algorithm which is polynomial both in the size of the input and in $1/\varepsilon$. The class of problems admitting an fptas is denoted by *FPTAS*.

## 3 General approximation schemes

We establish now general results giving necessary and sufficient conditions for the existence of fptas's for min-max (max-min) and min-max regret versions of problems $\mathcal{P}$ in $\mathcal{C}$. The sufficient conditions give rise to general approximation schemes.

### 3.1 Min-max and max-min

Investigating MIN-MAX $\mathcal{P}$ and MAX-MIN $\mathcal{P}$, we provide necessary and sufficient conditions for the existence of an fptas. Quite interestingly, we show that better fptas's can be obtained for the min-max version.

We first establish a general scheme that is valid for both cases. It is based on an *approximate binary search* which is a technique used to obtain fptas's for combinatorial optimization problems [7, 8, 18]. Let $V$ be a given value and $\varepsilon > 0$ be fixed. The approximate binary search is based on a testing procedure that outputs a positive or a negative answer; if it is positive then the optimal value *opt* verifies $opt \geq V$ and if it is negative then $opt < V(1+\varepsilon)$.

**Theorem 1** MAX-MIN $\mathcal{P}$ *(resp.* MIN-MAX $\mathcal{P}$*) is in* FPTAS *if and only if there exists an algorithm that finds for any instance $I$ of* MAX-MIN $\mathcal{P}$ *(resp.* MIN-MAX $\mathcal{P}$*) an optimal solution in time $r(|I|, c_{max}(I))$, where $r$ is a non-decreasing polynomial and $c_{max}(I) = \max_{i,s} c_i^s$.*

*The running time of the fptas is $O(\log \frac{\log m \cdot c_{max}(I)}{\log(1+\varepsilon)} r(|I|, \frac{t}{\varepsilon}))$, where $m$ is the number of coefficients in the objective function and $t$ is an upper bound of the size of any feasible solution of $I$.*

**Proof:** ($\Leftarrow$) Consider an instance $I$ of MAX-MIN $\mathcal{P}$ defined on a scenario set $S$ where each scenario $s \in S$ is represented by a vector $(c_1^s, \ldots, c_m^s)$. We denote by $A$ the algorithm that solves $I$ in time $r(|I|, c_{max}(I))$. Let $L_0$ and $U_0$ be a lower bound and an upper bound of the optimal value *opt(I)*, and $V$ a given value such that $L_0 < V < U_0$. The approximate binary search is used to tighten the bounds by either increasing the lower bound to $V$ or decreasing the upper bound to $V(1+\varepsilon)$ and continue this way until the ratio between the current upper and lower bounds falls below $1+\varepsilon$.

In order to do this, given $V$ and $\varepsilon > 0$, we compute an optimal solution of a simplified instance $I'$ of the same problem by rounding $c_i^s$ to $c_i'^s = \lfloor \frac{tc_i^s}{\varepsilon V} \rfloor$, where $t$ is an upper bound of the size of any feasible solution of $I$. However, applying algorithm $A$ to $I'$ can be very time consuming. We need to construct a more simplified instance $I''$ of MAX-MIN $\mathcal{P}$ defined as follows: $c_i''^s = c_i'^s$ if $c_i'^s < \frac{t}{\varepsilon}$ and $c_i''^s = \frac{t}{\varepsilon}$ otherwise.

We show in the following that if $opt(I'') \geq \frac{t}{\varepsilon}$ then $opt(I) \geq V$ and if $opt(I'') < \frac{t}{\varepsilon}$ then $opt(I) < V(1+\varepsilon)$.

First, suppose that $opt(I'') \geq \frac{t}{\varepsilon}$ and let $x''$ denote an optimal solution of $I''$ and $x^*$ denote an optimal solution of $I$. Then, the following inequalities hold:

$$opt(I) = \min_{s \in S} val(x^*, s) \geq \min_{s \in S} val(x'', s) \geq \frac{\varepsilon V}{t} \min_{s \in S} val'(x'', s) \geq \frac{\varepsilon V}{t} \min_{s \in S} val''(x'', s) \geq V$$

where $val'(x, s)$ and $val''(x, s)$ denote respectively the value of a solution $x \in X$, under scenario $s \in S$, for instance $I'$ and instance $I''$.

Suppose now that $opt(I'') < \frac{t}{\varepsilon}$. Then, for any solution $\widetilde{x}$, there exists a scenario $\widetilde{s}$ such that $val''(\widetilde{x}, \widetilde{s}) < \frac{t}{\varepsilon}$. This implies that $c_i''^{\widetilde{s}} < \frac{t}{\varepsilon}$ for any $i$ such that $\widetilde{x}_i = 1$. Therefore, we have $val'(\widetilde{x}, \widetilde{s}) = val''(\widetilde{x}, \widetilde{s})$, which gives $opt(I') < \frac{t}{\varepsilon}$.

We show now that $opt(I') < \frac{t}{\varepsilon}$ implies that $opt(I) < V(1+\varepsilon)$. Since $c_i'^s = \lfloor \frac{tc_i^s}{\varepsilon V} \rfloor$, we have

$$c_i^s < \frac{\varepsilon V}{t}(c_i'^s + 1), \text{ for all } s \in S.$$

Then an optimal solution $x^*$ for $I$ satisfies $val(x^*, s) < \frac{\varepsilon V}{t} val'(x^*, s) + \varepsilon V$, for all $s \in S$, which implies that

$$opt(I) = \min_{s \in S} val(x^*, s) < \frac{\varepsilon V}{t} \min_{s \in S} val'(x^*, s) + \varepsilon V < \frac{\varepsilon V}{t} opt(I') + \varepsilon V < V(1+\varepsilon).$$

Instead of using a standard binary search with $V = \frac{L+U}{2}$, we can use an accelerated version of the approximate binary search by setting iteratively $V = \sqrt{LU}$, as suggested in [7]. More precisely, since computing the exact value $\sqrt{LU}$ can be time consuming, it is shown in [7] that an approximate value of $\sqrt{LU}$ can be computed without affecting the time complexity. In our case, using this accelerated version requires one to have $L_0 \neq 0$. In order to detect if instance $I$ has $opt(I) = 0$, we construct an instance $\overline{I}$ of MAX-MIN $\mathcal{P}$, where $\overline{c}_i^s = c_i^s$ if $c_i^s = 0$ and $\overline{c}_i^s = 1$ otherwise. Clearly, $opt(I) = 0$ if and only if $opt(\overline{I}) = 0$. If applying algorithm $A$

to $\overline{I}$ we obtain $opt(\overline{I}) = 0$ then the same solution is an optimal solution for $I$. Otherwise, we can start the accelerate approximate binary search with $L_0 = 1$.

The number of tests for obtaining a lower bound and an upper bound $L$ and $U$ such that $\frac{U}{L} \leq 1 + \varepsilon$ is $O(\log \frac{\log \frac{U_0}{L_0}}{\log(1+\varepsilon)})$) (see [7] for more details). Each test requires solving an instance $I''$ in $r(|I''|, \frac{t}{\varepsilon})$ time. We obtain a total $O(\log \frac{\log m \cdot c_{max}(I)}{\log(1+\varepsilon)} r(|I|, \frac{t}{\varepsilon}))$.

($\Rightarrow$) Consider $I$ an instance of MAX-MIN $\mathcal{P}$. Applying the fptas to instance $I$ with $\varepsilon_0 = 1/(m \cdot c_{max}(I) + 1)$ returns a solution whose value $val_0$ has the property $opt(I) - val_0 \leq (1 + \varepsilon_0)val_0 - val_0 = \varepsilon_0 val_0 < 1$. Since $val_0$ and $opt(I)$ are integers, we get $val_0 = opt(I)$. Moreover, the time of the algorithm is polynomial in $|I|$ and $c_{max}(I)$.

The proof can be easily adapted to MIN-MAX $\mathcal{P}$. □

The sufficient part of Theorem 1 gives a first general approximation scheme. However, one can often obtain a faster fptas by stopping the approximate binary search as soon as we get polynomially related upper and lower bounds and then applying the scaling technique only once.

**Proposition 1** *Assuming that any instance $I$ of* MAX-MIN $\mathcal{P}$ *(resp.* MIN-MAX $\mathcal{P}$*) can be solved in time $r(|I|, c_{max}(I))$, where $r$ is a non-decreasing polynomial and $c_{max}(I) = \max_{i,s} c_i^s$, if a lower bound and an upper bound $L$ and $U$ of $opt(I)$ are given such that $U \leq q(|I|)L$, where $q$ is a non-decreasing polynomial, then an $(1 + \varepsilon)$-approximate solution can be found in time $r(|I|, \frac{t}{\varepsilon}q(|I|))$, where $t$ is an upper bound of the size of any feasible solution of $I$.*

**Proof:** Consider $\mathcal{P}$ a maximization problem and an instance $I$ of MAX-MIN $\mathcal{P}$ defined on a scenario set $S$ where each scenario $s \in S$ is represented by a vector $(c_1^s, \ldots, c_m^s)$. Let $\overline{I}$ be the instance of MAX-MIN $\mathcal{P}$ derived from $I$ where each scenario $s \in S$ is represented by a vector $(\overline{c}_1^s, \ldots, \overline{c}_m^s)$, with $\overline{c}_i^s = \lfloor \frac{tc_i^s}{\varepsilon L} \rfloor$. Let $x^*$ and $\overline{x}^*$ denote respectively an optimal solution of instance $I$ and instance $\overline{I}$. Let $\overline{val}(x, s)$ denote the value of a solution $x$ in scenario $s$ for $\overline{I}$. We have

$$\frac{\varepsilon L}{t}\overline{c}_i^s \leqslant c_i^s < \frac{\varepsilon L}{t}(\overline{c}_i^s + 1), \text{ for all } s \in S,$$

and thus,
$$\tfrac{\varepsilon L}{t}\overline{val}(\overline{x}^*, s) \leqslant val(\overline{x}^*, s) < \tfrac{\varepsilon L}{t}\overline{val}(\overline{x}^*, s) + \varepsilon L, \text{ for all } s \in S,$$

which implies $\min_{s \in S} val(\overline{x}^*, s) \geqslant \frac{\varepsilon L}{t} \min_{s \in S} \overline{val}(\overline{x}^*, s)$.

Since $\overline{x}^*$ is an optimal solution in $\overline{I}$, we have

$$opt(\overline{I}) = \min_{s \in S} \overline{val}(\overline{x}^*, s) \geqslant \min_{s \in S} \overline{val}(x^*, s)$$

and thus, the value of an optimal solution of $\overline{I}$ has, in $I$, the value

$$\min_{s \in S} val(\overline{x}^*, s) \geqslant \frac{\varepsilon L}{t} \min_{s \in S} \overline{val}(\overline{x}^*, s) \geqslant \frac{\varepsilon L}{t} \min_{s \in S} \overline{val}(x^*, s) > \min_{s \in S} val(x^*, s) - \varepsilon L \geqslant opt(I)(1 - \varepsilon).$$

The proof can be easily adapted to MIN-MAX $\mathcal{P}$. □

For the min-max version, we can compute polynomially related upper and lower bounds directly, i.e. without resorting to approximate binary search. This allows us to obtain fast fptas's by solving only one scaled instance using Proposition 1.

**Proposition 2** *If a minimization problem $\mathcal{P}$ is $f(n)$-approximable in time $p(n)$, where $p$ and $f$ are polynomials, then for any instance $I$ of* Min-Max $\mathcal{P}$ *defined on a set of $k$ scenarios, there exist a lower bound and an upper bound $L$ and $U$ of $opt(I)$ computable in time $p(|I|)$ such that $U \leq kf(|I|)L$.*

**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 $(c_1^s, \ldots, c_m^s)$ 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 $c'_i = \sum_{s=1}^{k} \frac{c_i^s}{k}$, $i = 1, \ldots, m$. Let $\widetilde{x}$ be an $f(|I|)$-approximate solution of $I'$. Thus we have $\sum_{s \in S} \frac{1}{k} val(\widetilde{x}, s) \leq f(|I|) \min_{x \in X} \sum_{s \in S} \frac{1}{k} val(x, s)$. Clearly $U = \max_{s \in S} val(\widetilde{x}, s)$ is an upper bound. Moreover $L = \frac{1}{f(|I|)} \sum_{s \in S} \frac{1}{k} val(\widetilde{x}, s)$ is a lower bound since

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

Finally, we have $U = \max_{s \in S} val(\widetilde{x}, s) \leq \sum_{s \in S} val(\widetilde{x}, s) = k \sum_{s \in S} \frac{1}{k} val(\widetilde{x}, s) = kf(|I|)L$. □

The condition in Proposition 2 is not restrictive, since if $\mathcal{P}$ is not approximable, we cannot hope to obtain an fptas for Min-Max $\mathcal{P}$. However, the following more restrictive corollary may prove useful.

**Corollary 1** *If a minimization problem $\mathcal{P}$ is solvable in time $p(n)$, where $p$ is a polynomial, then for any instance $I$ of* Min-Max $\mathcal{P}$ *defined on a set of $k$ scenarios, there exist a lower bound and an upper bound $L$ and $U$ of $opt(I)$ computable in time $p(|I|)$ such that $U \leq kL$.*

For the max-min version, however, constructing $L$ and $U$ as before with $L = \min_{s \in S} val(\widetilde{x}, s)$ and $U = f(|I|) \sum_{s \in S} \frac{1}{k} val(\widetilde{x}, s)$, where $\widetilde{x}$ is an $f(|I|)$-approximate solution of $I'$, does not allow us to bound the ratio $\frac{U}{L}$. In particular, for the knapsack problem, this ratio can be exponential in the size of the input, as can be seen from the following simple example. Consider an instance with two scenarios, $n$ items with weights $w_i = 1$, profits $p_i^1 = 1$ and $p_i^2 = 2^n - 1$, $i = 1, \ldots, n$, and a capacity $b = 1$. Computing $L$ and $U$ as before yields $L = 1$ and $U = 2^{n-1}$.

### 3.2 Min-max regret

We prove first that min-max regret versions of *NP*-hard problems are not at all approximable.

**Proposition 3** *Given an NP-hard problem $\mathcal{P}$, for any function $f : \mathbb{N} \to (1, \infty)$,* Min-Max Regret $\mathcal{P}$ *is not $f(n)$-approximable even for two scenarios, unless $P = NP$.*

**Proof:** We construct a polynomial reduction from $\mathcal{P}$ to Min-Max Regret $\mathcal{P}$. Consider an instance $I$ of $\mathcal{P}$ on $m$ variables where $X$ is the set of feasible solutions and $c_i$ is the coefficient of variable $x_i$ in the objective function, $i = 1, \ldots, m$. We define an instance $I'$ of Min-Max Regret $\mathcal{P}$ on two scenarios $s_1$ and $s_2$ on the same set of $m$ variables and same set of feasible solutions $X$. The coefficients of variable $x_i$ in scenarios $s_1$ and $s_2$ are $c_i$, $i = 1, \ldots, m$. Clearly a solution is an optimal solution for $I$ of value $opt(I)$ if and only if it is also an optimal

solution for $I'$ of value 0. Suppose now that MIN-MAX REGRET $\mathcal{P}$ is $f(n)$-approximable for a given function $f : \mathbb{N} \to (1, \infty)$. Applying this $f(n)$-approximation algorithm on $I'$, we obtain an optimal solution for $I'$, and thus we can obtain in polynomial time an optimum solution for $I$. □

Therefore, in the following we consider only min-max regret versions of polynomial-time solvable problems.

**Proposition 4** *If problem $\mathcal{P}$ is solvable in time $p(n)$, where $p$ is a polynomial, then for any instance $I$ of* MIN-MAX REGRET $\mathcal{P}$ *defined on a set of $k$ scenarios, there exist a lower bound and an upper bound $L$ and $U$ of $opt(I)$ computable in time $p(|I|)$ such that $U \leq kL$.*

**Proof:** Consider $\mathcal{P}$ a minimization problem and an instance $I$ of MIN-MAX REGRET $\mathcal{P}$ defined on a set $S$ of $k$ scenarios where each scenario $s \in S$ is represented by $(c_1^s, \ldots, c_m^s)$, 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 $c'_i = \sum_{s=1}^{k} \frac{c_i^s}{k}$, $i = 1, \ldots, m$. Let $x^*$ be an optimal solution of $I'$. Clearly $U = \max_{s \in S}(val(x^*, s) - val_s^*)$ is an upper bound of $opt(I)$. Moreover $L = \sum_{s \in S} \frac{1}{k}(val(x^*, s) - val_s^*)$ is a lower bound of $opt(I)$ since

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

Finally, we have $U = \max_{s \in S}(val(x^*, s) - val_s^*) \leq \sum_{s \in S}(val(x^*, s) - val_s^*) = kL$.

Consider now $\mathcal{P}$ a maximization problem and an instance $I$ of MIN-MAX REGRET $\mathcal{P}$ defined on a set $S$ of $k$ scenarios. We can show as before that the bounds $L = \sum_{s \in S} \frac{1}{k}(val_s^* - val(x^*, s))$ and $U = \max_{s \in S}(val_s^* - val(x^*, s))$ satisfy $U \leq kL$. □

We can now provide a necessary and sufficient condition for obtaining fptas's for MIN-MAX REGRET $\mathcal{P}$.

**Theorem 2** *Given a polynomial-time solvable problem $\mathcal{P}$,* MIN-MAX REGRET $\mathcal{P}$ *is in FPTAS if and only if there exists an algorithm that finds for any instance $I$ of* MIN-MAX REGRET $\mathcal{P}$ *an optimal solution in time $r(|I|, U)$, where $r$ is a non-decreasing polynomial and $U$ is an upper bound of $opt(I)$, such that $U \leq kL$, $L$ being a lower bound of $opt(I)$.*

*If $\mathcal{P}$ is solvable in time $p(n)$, the running time of the fptas is $(k+1)p(|I|) + r(|I|, \frac{2tk}{\varepsilon} + t)$ where $k$ is the number of scenarios and $t$ is an upper bound of the size of any feasible solution of $I$.*

**Proof:** ($\Leftarrow$) Consider $\mathcal{P}$ a minimization problem and an instance $I$ of MIN-MAX REGRET $\mathcal{P}$ defined on a scenario set $S$ where each scenario $s \in S$ is represented by a vector $(c_1^s, \ldots, c_m^s)$.

Let $\overline{I}$ denote the instance derived from $I$, by scaling each entry $c_i^s$ as follows: $\overline{c}_i^s = \lfloor \frac{2tc_i^s}{\varepsilon L} \rfloor$. Let $x^*$ and $\overline{x}^*$ denote respectively an optimal solution of instance $I$ and instance $\overline{I}$ and let $x_s^*$, $\overline{x}_s^*$ denote respectively, an optimal solution of instance $I$ and $\overline{I}$ restricted to scenario $s$.

Then, we have, for all $s \in S$,

$$\begin{aligned} val(\overline{x}^*, s) - val(x_s^*, s) &< \frac{\varepsilon L}{2t} \overline{val}(\overline{x}^*, s) - val(x_s^*, s) + \frac{\varepsilon}{2} L \\ &\leq \frac{\varepsilon L}{2t} (\overline{val}(\overline{x}^*, s) - \overline{val}(x_s^*, s)) + \frac{\varepsilon}{2} L \\ &\leq \frac{\varepsilon L}{2t} (\overline{val}(\overline{x}^*, s) - \overline{val}(\overline{x}_s^*, s)) + \frac{\varepsilon}{2} L \end{aligned}$$

and thus

$$\begin{aligned}
\max_{s\in S}\{val(\overline{x}^*,s)-val(x^*_s,s)\} &< \max_{s\in S}\left\{\frac{\varepsilon L}{2t}(\overline{val}(\overline{x}^*,s)-\overline{val}(\overline{x}^*_s,s))\right\}+\frac{\varepsilon}{2}L\\
&\leq \max_{s\in S}\left\{\frac{\varepsilon L}{2t}(\overline{val}(x^*,s)-\overline{val}(\overline{x}^*_s,s))\right\}+\frac{\varepsilon}{2}L\\
&\leq \max_{s\in S}\{val(x^*,s)-val(x^*_s,s)+val(x^*_s,s)-\frac{\varepsilon L}{2t}\overline{val}(\overline{x}^*_s,s)\}+\frac{\varepsilon}{2}L\\
&\leq \max_{s\in S}\{val(x^*,s)-val(x^*_s,s)+val(\overline{x}^*_s,s)-\frac{\varepsilon L}{2t}\overline{val}(\overline{x}^*_s,s)\}+\frac{\varepsilon}{2}L\\
&< \max_{s\in S}\{val(x^*,s)-val(x^*_s,s)\}+\varepsilon L\leq opt(I)(1+\varepsilon)
\end{aligned}$$

We show in the following that such a solution $\overline{x}^*$ of instance $\overline{I}$ for Min-Max Regret $\mathcal{P}$ can be obtained in polynomial time in $|I|$ and $\frac{1}{\varepsilon}$. The bounds $L$ and $U$ can be computed in time $p(|I|)$ by Proposition 4. In order to compute an optimal solution for $\overline{I}$, we apply the algorithm (which exists by hypothesis) that runs in time $r(|\overline{I}|,U(\overline{I}))$.

Computing optimal values on each scenario and bounds $L$ and $U$ requires solving $k+1$ instances of problem $\mathcal{P}$. Since $opt(\overline{I})\leq\frac{2topt(I)}{\varepsilon L}+t\leq\frac{2tU}{\varepsilon L}+t\leq\frac{2tk}{\varepsilon}+t$, and $r$ is non-decreasing, the total time for computing the $(1+\varepsilon)$-approximation is $(k+1)p(|I|)+r(|\overline{I}|,U(\overline{I}))\leq(k+1)p(|I|)+r(|I|,\frac{2tk}{\varepsilon}+t)$.

($\Rightarrow$) Consider $\mathcal{P}$ a minimization problem, and let $I$ be an instance of Min-Max Regret $\mathcal{P}$. Applying the fptas for instance $I$ with $\varepsilon_0=1/(U+1)$, returns an optimal solution in time polynomial in $|I|$ and $U+1$.

The proof can be easily adapted to Min-Max Regret $\mathcal{P}$ where $\mathcal{P}$ is a maximization problem. □

### 3.3 General remarks

It is well known that the existence of a pseudo-polynomial algorithm, that is an algorithm that runs in polynomial time in the size of the input and the largest value in the instance, is not a sufficient condition for the existence of an fptas. General subclasses of pseudo-polynomial algorithms were investigated previously for standard combinatorial optimization problems. In particular, Pruhs and Woeginger [16] identify the same subclass as in Theorem 1, dedicated to min-max (max-min) versions. However, their proof, which relies on a complete ranking of decision variables $x_i$ according to values $c_i$, cannot be extended to the min-max (max-min) case where values $c_i^1,\ldots,c_i^{|S|}$ associated to variables $x_i$ only lead to a partial ranking of these variables. In [19], the existence of fptas's is proved for standard combinatorial optimization problems which admit dynamic programming formulations verifying specific conditions. Even if this result is quite interesting, it cannot be applied to the large variety of problems, including Spanning Tree and Weighted Perfect Matching, which are not known to admit such formulations.

Theorems 1 and 2 identify two subclasses of pseudo-polynomial algorithms that provide necessary and sufficient conditions for the existence of an fptas for min-max and min-max regret versions respectively.

For the min-max (max-min) version, the condition is related to the existence of an algorithm polynomial in $c_{max}(I)$, the largest value of the coefficients in the objective function.

Observe that $U_0 = m.c_{max}(I)$ is a trivial upper bound of the optimal value in the min-max (max-min) version. Thus, for any upper bound $U \leq U_0$, an algorithm polynomial in $U$ is also polynomial in $c_{max}(I)$.

For the min-max regret version, the existence of an fptas requires stronger conditions than the min-max (max-min) version despite the similarity of these problems. Even if there exists an exact algorithm polynomial in the largest value in the objective function $c_{max}(I)$, it cannot be transformed into an fptas in general. This is illustrated by Min-Max Regret Knapsack, which admits such an algorithm, but does not admit an fptas as a consequence of Proposition 3.

Min-max and min-max regret versions of some problems, like shortest path, admit pseudo-polynomial algorithms based on dynamic programming [11]. For some dynamic programming formulations, we can easily obtain algorithms polynomial in the size of the instance and in $U$, by discarding partial solutions with value more than $U$ on at least one scenario. We illustrate this approach in sections 4.1 and 4.4 for the shortest path and knapsack problems.

For other problems, which are not known to admit pseudo-polynomial algorithms based on dynamic programming, specific algorithms polynomial in the size of the instance and in $U$ are required. We present such algorithms for Min-Max Spanning Tree (section 4.2) and Min-Max Weighted Perfect Matching in planar graphs (section 4.3).

Unfortunately, these algorithms cannot be adapted directly in order to obtain algorithms satisfying Theorem 2 for min-max regret versions. The basic difficulty here is that, if we can find an algorithm in $r(|I|, U(I))$ for any instance $I$ of Min-Max $\mathcal{P}$, the direct extension of this algorithm for the corresponding instance $I'$ of Min-Max Regret $\mathcal{P}$ will be in $r(|I'|, U(I') + opt_{max})$ where $opt_{max} = \max_{s \in S} val_s^*$ is a value which is not necessarily polynomially related to $U(I')$.

However, for problems whose feasible solutions have a fixed size, such as spanning tree and perfect matching problems, we reduced the min-max regret version to a min-max version in [2]. In this context, we need to consider instances where some coefficients are negative and possibly non-integral but such that any feasible solution has a non-negative integral value. For an optimization problem $\mathcal{P}$, we denote by $\mathcal{P}'$ the extension of $\mathcal{P}$ to these instances. More precisely, we proved the following result.

**Proposition 5 ([2])** *For any polynomial-time solvable minimization problem $\mathcal{P}$ whose feasible solutions have a fixed size and for any function $f : \mathbb{N} \to (1, \infty)$, if* Min-Max $\mathcal{P}'$ *is $f(n)$-approximable in time $p(n)$, where $p$ is a polynomial, then* Min-Max Regret $\mathcal{P}$ *is $f(n)$-approximable in time $p(n)$.*

Proposition 5 can be adapted to handle the min-max regret version of maximization problems.

# 4 Applications

In this section, we apply the previous results to min-max and min-max regret versions of shortest path, minimum spanning tree, minimum weighted perfect matching in planar graphs, and knapsack problems with a constant number $k$ of scenarios. We also compare the running time for our algorithms and for the fptas obtained using an approximation of the non-dominated set, and show a significant improvement.

### 4.1 Shortest Path

In [11], Kouvelis and Yu proved the *NP*-hardness of min-max and min-max regret versions of shortest path, even for two scenarios.

Consider an instance $I$ of Min-Max (Regret) Shortest Path defined by a directed graph $G = (V, A)$, with $V = \{1, \ldots, n\}$ and $|A| = m$, and a set $S$ of $k$ scenarios giving for each arc $(i, j) \in A$ its cost $c_{ij}^s$ under scenario $s$. Denote by $c_{ij}$ the vector of size $k$ formed by $c_{ij}^s$, $s \in S$. We are interested in optimal paths from 1 to $n$.

We give now pseudo-polynomial algorithms satisfying Proposition 1 (and respectively Theorem 2) for Min-Max Shortest Path (and respectively Min-Max Regret Shortest Path).

**Proposition 6** *Given $U$ an upper bound on the optimal value, then* Min-Max Shortest Path *and* Min-Max Regret Shortest Path *can be solved in time $O(n^2U^{k-1})$.*

**Proof:** For Min-Max Shortest Path, the algorithm aims at generating candidate vectors $(v_1, \ldots, v_k)$ corresponding to feasible paths from 1 to $n$ whose value under scenario $s$ is $v_s$, $s = 1, \ldots, k$. Since we know an upper bound $U$, we can restrict to vectors such that $v_s \leq U$, $s = 1, \ldots, k$. Moreover, a min-max solution will necessarily correspond to a non-dominated vector, which means that the number of candidate vectors is in $O(U^{k-1})$. Since checking non-dominance is computationally costly, we shall actually consider a superset of the set of non-dominated vectors which is easier to determine. This superset consists of all possible vectors $(v_1, \ldots, v_k)$ with $v_s \leq U$, $s = 1, \ldots, k$, such that for every possible configuration $(v_1, \ldots, v_{k-1})$ we only retain the vector $(v_1, \ldots, v_{k-1}, v_k)$ with the smallest $v_k$ value. We observe that the cardinality of this superset is also in $O(U^{k-1})$. Once this set is determined, we scan all of its vectors in order to select one which minimizes $\max_{s=1,\ldots,k} v_s$.

A possible implementation for determining this superset progressively updates a $(k-1)$-dimensional matrix $M(v_1, \ldots, v_{k-1})$, with $v_s \in \{0, \ldots, U\}$, $s = 1, \ldots, k-1$. Each entry of $M$ contains an $n$-dimensional vector indexed by $i$ from 1 to $n$, which stores the smallest value $v_k$ for a path from 1 to $i$ whose values on the first $k-1$ scenarios are $v_1, \ldots, v_{k-1}$ (as well as the index of the previous node in the path if we wish to exhibit a corresponding path). All the entries of $M$ are initialized to $U + 1$, except for $M(0, \ldots, 0)(1)$ which is set to 0.

The algorithm scans $M$ in lexicographic order. For a given entry $(v_1, \ldots, v_{k-1})$, it selects the unvisited node $i$ with the smallest value $v_k$. If $v_k \leq U$ then we update $M$ considering all arcs $(i, j)$ in $A$: for each arc $(i, j)$, $M(v_1 + c_{ij}^1, \ldots, v_{k-1} + c_{ij}^{k-1})(j)$ is updated with the smallest value between its previous value and $v_k + c_{ij}^k$. This algorithm requires $O(n^2U^{k-1})$ total time for selection and $O(mU^{k-1})$ total time for update. Therefore, its running time is in $O(n^2U^{k-1})$.

Consider now Min-Max Regret Shortest Path. Let $(val_s^*)^i$, $s \in S$, $i = 1, \ldots, n$, be the value of a shortest path in graph $G$ from 1 to $i$ under scenario $s$.

We describe a similar algorithm that computes all possible vectors $(r_1, \ldots, r_k)$ of regrets such that $r_s \leq U$, $s = 1, \ldots, k$, corresponding to paths from 1 to $n$. For each possible configuration $(r_1, \ldots, r_{k-1})$, only the vector $(r_1, \ldots, r_k)$ with the smallest regret $r_k$ is retained. As before, the resulting set of cardinality $O(U^k)$ is scanned in order to detect one which minimizes $\max_{s=1,\ldots,k} r_s$.

As for the min-max case, the implementation progressively updates a regret matrix $M'$ with the same structure as $M$. The selection step is also based on a lexicographic scanning

of $M'$. The only difference is the way of updating regret vectors. Consider arc $(i,j) \in A$ and let $P_i$ be a path in $G$ from 1 to $i$ of regret $r_s^i = val(P_i,s) - (val_s^*)^i$, $s \in S$. Denote by $P_j$ the path constructed from $P_i$ by adding arc $(i,j)$. The regret of $P_j$ is $r_s^j = val(P_i,s) + c_{ij}^s - (val_s^*)^j = r_s^i + (val_s^*)^i + c_{ij}^s - (val_s^*)^j$, $s = 1,\ldots,k$. Observe that $(val_s^*)^i + c_{ij}^s - (val_s^*)^j \geq 0$, $s = 1,\ldots,k$ justifies the lexicographic scanning of $M'$, without backtracking. Thus, once a node is selected from an entry $M'(r_1,\ldots,r_{k-1})$, for each arc $(i,j)$, we update $M'(r_1 + (val_1^*)^i + c_{ij}^1 - (val_1^*)^j, \ldots, r_{k-1} + (val_{k-1}^*)^i + c_{ij}^{k-1} - (val_{k-1}^*)^j)$ with the smallest value between its previous value and $r_k + (val_k^*)^i + c_{ij}^k - (val_k^*)^j$. The running time of the algorithm is the same as for the min-max version, i.e. $O(n^2U^{k-1})$. □

**Corollary 2** Min-Max Shortest Path *admits an fptas running in time* $O(\frac{n^{k+1}}{\varepsilon^{k-1}})$.

**Proof:** This results from Corollary 1, Proposition 6, and Proposition 1. □

**Corollary 3** Min-Max Regret Shortest Path *admits an fptas running in time* $O(\frac{n^{k+1}}{\varepsilon^{k-1}})$.

**Proof:** This results from Proposition 4, Proposition 6, and Theorem 2. □

Warburton describes in [18] an fptas for approximating the non-dominated set for the multi-objective version of the shortest path problem. From this fptas, Warburton derives an fptas for Min-Max Shortest Path in acyclic graphs with running time $O(\frac{n^{2k+1}}{\varepsilon^{2k-2}})$, whereas our running time, for general graphs, is better.

### 4.2 Minimum Spanning Tree

In [11], Kouvelis and Yu proved the *NP*-hardness of min-max and min-max regret versions of minimum spanning tree, even for two scenarios. We first describe algorithms for Min-Max Spanning Tree with running time polynomial in a suitably chosen upper bound on the optimal value.

Consider an instance of Min-Max (Regret) Spanning Tree represented by a graph $G = (V,E)$ where $|V| = n$, $|E| = m$, $c_{ij}^s$ is the cost of edge $(i,j)$ in scenario $s \in S$ and $|S| = k$.

**Proposition 7** *Given $U$ an upper bound on the optimal value, then* Min-Max Spanning Tree *can be solved in time* $O(mn^4U^k \log U)$.

**Proof:** We can solve Min-Max Spanning Tree using an extension of the matrix tree theorem to the multiple scenarios case as presented in appendix A.

The optimal value *opt* of Min-Max Spanning Tree can be computed by considering, for each monomial in (2), the largest power $v_{max} = \max_{s=1,\ldots,k} v_s$. The minimum value of $v_{max}$ over all monomials corresponds to *opt*.

Actually, instead of computing all monomials, we can use, as suggested in [8], the algorithm presented in [13]. When applied to matrix $A_r(y_1,\ldots,y_k)$, this algorithm can compute the determinant polynomial up to a specified degree in each variable in opposition to the classical method of Edmonds [5]. In this case, it is sufficient to compute the polynomial determinant up to degree $U$ in each variable $y_s$ for $s = 1,\ldots,k$. The algorithm in [13] requires $O(n^4)$ multiplications and additions of polynomials. The time needed to multiply two multivariate polynomials of maximum degree $d_s$ in variable $y_s$ for $s = 1,\ldots,k$ is $\prod_{s=1}^k d_s \log \prod_{s=1}^k d_s$ [1]. Thus, the running time to compute the polynomial determinant is $O(n^4U^k \log U)$.

Once an optimal vector is identified, a corresponding spanning tree can be constructed using self-reducibility [14]. It consists of testing iteratively for each edge, if the graph obtained by contracting this edge admits a spanning tree of the required vector of adjusted values on all scenarios (subtracting iteratively from the required vector of values the vector of costs $c_{ij}^s$, $s \in S$, for each edge $(i,j)$ being tested). In at most $m-(n-1)$ iterations such a spanning tree is obtained. Hence, the self-reducibility requires $O(m)$ computations of determinant polynomial. □

**Corollary 4** MIN-MAX SPANNING TREE *admits an fptas running in time* $O(\frac{mn^{k+4}}{\varepsilon^k}\log\frac{n}{\varepsilon})$.

**Proof:** This results from Corollary 1, Proposition 7, and Proposition 1. □

**Corollary 5** MIN-MAX REGRET SPANNING TREE *admits an fptas running in time* $O(\frac{mn^{k+4}}{\varepsilon^k}\log\frac{n}{\varepsilon})$.

**Proof:** Notice that Corollary 1, Proposition 7, and Proposition 1 remain true even for the instances of spanning tree where some coefficients are negative but any feasible solution has a non-negative value. Thus, MIN-MAX SPANNING TREE′ is in FPTAS. The result follows from Proposition 5. □

The obtained fptas's for MIN-MAX (REGRET) SPANNING TREE are at least as good as, and usually better than, those derived from multi-objective approximation schemes. Indeed, the running time of the fptas obtained in [2] using the general multi-objective approximation scheme presented in [15] is $O(\frac{n^{k+4}}{\varepsilon^{2k}}(\log n.c_{max}(I))^k\log\frac{n}{\varepsilon})$ to identify a vector. In order to obtain a corresponding solution, we resort as before to self-reducibility, which leads to a total time $O(\frac{n^{k+4}}{\varepsilon^{2k}}(\log n.c_{max}(I))^k\log\frac{n}{\varepsilon}+\frac{mn^{k+4}}{\varepsilon^k}\log\frac{n}{\varepsilon})$.

### 4.3 Minimum Weighted Perfect Matching in planar graphs

In this section we first state the complexity of min-max and min-max regret versions of minimum weighted perfect matching in planar graphs.

Consider now an instance of MIN-MAX (REGRET) WEIGHTED PERFECT MATCHING defined on a planar graph $G=(V,E)$ with $|V|=2n$, $|E|=m$; $c_{ij}^s$ is the weight of edge $(i,j)$ in scenario $s\in S$ and $|S|=k$.

**Theorem 3** MIN-MAX (REGRET) WEIGHTED PERFECT MATCHING *is NP-hard even for two scenarios and planar graphs.*

**Proof:** See appendix B. □

We give now an algorithm polynomial in the size of the input and an upper bound of the optimal value for MIN-MAX WEIGHTED PERFECT MATCHING.

**Proposition 8** *Given $U$ an upper bound on the optimal value, then* MIN-MAX WEIGHTED PERFECT MATCHING *in planar graphs can be solved in time* $O(mn^4U^k\log U)$.

**Proof:** We can solve MIN-MAX WEIGHTED PERFECT MATCHING in planar graphs by adapting the result of Pfaffian orientations to the multiple scenarios case as presented in appendix A. The algorithm described in [12] can compute the Pfaffian polynomial of $B(y_1,\ldots,y_n)$

up to a specified degree in each variable. The running time to compute the optimal value is the same as for spanning tree. An optimal solution can be constructed by self-reducibility in $O(m)$ calls to the procedure of Pfaffian polynomial computation. □

**Corollary 6** MIN-MAX WEIGHTED PERFECT MATCHING *in planar graphs admits an fptas running in time* $O(\frac{mn^{k+4}}{\varepsilon^k}\log\frac{n}{\varepsilon})$.

**Proof:** This results from Corollary 1, Proposition 8, and Proposition 1. □

**Corollary 7** MIN-MAX REGRET WEIGHTED PERFECT MATCHING *in planar graphs admits an fptas running in time* $O(\frac{mn^{k+4}}{\varepsilon^k}\log\frac{n}{\varepsilon})$.

**Proof:** Observing that Corollary 1, Proposition 8, and Proposition 1 remain true even for the instances of MIN-MAX WEIGHTED PERFECT MATCHING′, we can apply Proposition 5 to obtain an fptas. □

Here again, the obtained fptas's for MIN-MAX (REGRET) WEIGHTED PERFECT MATCHING in planar graphs are at least as good as, and usually better than, those derived from multi-objective approximation schemes. Indeed, the running time of the fptas obtained in [2] by applying the general scheme presented in [15] is $O(\frac{n^{k+4}}{\varepsilon^{2k}}(\log n.c_{max}(I))^k\log\frac{n}{\varepsilon})$ to identify a vector. In order to obtain a corresponding solution, we resort as before to self-reducibility, which leads to a total time $O(\frac{n^{k+4}}{\varepsilon^{2k}}(\log n.c_{max}(I))^k\log\frac{n}{\varepsilon}+\frac{mn^{k+4}}{\varepsilon^k}\log\frac{n}{\varepsilon})$.

Concerning MIN-MAX (REGRET) WEIGHTED PERFECT MATCHING for general graphs, the existence of an fptas remains an open question. With our approach, even if Proposition 2 (or respectively Proposition 4) is clearly satisfied, the existence of an algorithm polynomial in $c_{max}(I)$ (or respectively $U$) is open.

### 4.4 Knapsack

We describe in this section a pseudo-polynomial algorithm to solve MAX-MIN KNAPSACK. The standard procedure to solve the classical knapsack problem is based on dynamic programming. We give an extension to the multi-scenario case which is very similar to the procedure presented in [6] by Erlebach *et al.* to solve the multi-objective knapsack problem. This procedure is easier and more efficient than the one originally presented by Yu [20].

Consider an instance of MAX-MIN KNAPSACK where each item $i$ has a weight $w_i$ and profit $p_i^s$, for $i=1,\ldots,n$ and $s=1,\ldots,k$, and a capacity $b$.

**Proposition 9** *Given $U$ an upper bound on the optimal value, then* MAX-MIN KNAPSACK *can be solved in time* $O(nU^k)$.

**Proof:** Let $W_i(v_1,\ldots,v_k)$ denote the minimum weight of any subset of items among the first $i$ items with profit $v_s$ in each scenario $s\in S$. The initial condition of the algorithm is given by setting $W_0(0,\ldots,0)=0$ and $W_0(v_1,\ldots,v_k)=b+1$ for all other combinations. The recursive relation is given by the following
If $v_s\geq p_i^s$ for all $s\in S$, then

$$W_i(v_1,\ldots,v_k)=\min\{W_{i-1}(v_1,\ldots,v_k),W_{i-1}(v_1-p_i^1,\ldots,v_k-p_i^k)+w_i\}$$

else $W_i(v_1,\ldots,v_k) = W_{i-1}(v_1,\ldots,v_k)$

for $i = 1,\ldots,n$.

Each entry of $W_n$ satisfying $W_n(v_1,\ldots,v_k) \leq b$ corresponds to a feasible solution with profit $v_s$ in scenario $s$. The set of items leading to a feasible solution can be determined easily using standard bookkeeping techniques. The feasible solutions are collected and an optimal solution to MAX-MIN KNAPSACK is obtained by picking a feasible solution minimizing $\max_{s\in S} v_s$. Since $v_s \in \{0,\ldots,U\}$, the running time of this approach is given by going through the complete profit space for every item, and is $O(nU^k)$. □

The algorithm presented by Yu [20] has a running time $O(nbU^k)$.

**Corollary 8** MAX-MIN KNAPSACK *admits an fptas running in time* $O(n^{k+1}\log\log(n.c_{max}(I))+\frac{n^{k+1}}{\varepsilon^k})$.

**Proof:** This results from Proposition 9 and Theorem 1 and stopping the binary search used in Theorem 1 when $\frac{U}{L} \leq 2$. □

We can also obtain an fptas for MAX-MIN KNAPSACK using Proposition 9 and Theorem 1 and stopping the binary search used in Theorem 1 when $\frac{U}{L} \leq 1+\varepsilon$, but in this case the time of the fptas is $O(\frac{n^{k+1}}{\varepsilon^k}(\log\frac{\log(n.c_{max}(I))}{\log(1+\varepsilon)}))$, which is larger than the time of the previous fptas.

The running time of the fptas obtained in [2] by using the relationship between the max-min and multi-objective versions of knapsack is $O(\frac{n^{k+1}}{\varepsilon^k}(\log n.c_{max}(I))^k)$. Thus, in this paper, we obtain an fptas for MAX-MIN KNAPSACK with a better running time than the previous one.

Consider now an instance of MIN-MAX KNAPSACK where each item $i$ has a weight $w_i$ and cost $c_i^s$, for $i = 1,\ldots,n$ and $s = 1,\ldots,k$, and a required minimal total weight $b$.

We can adapt the proof of Proposition 9 obtaining a similar result.

**Proposition 10** *Given $U$ an upper bound on the optimal value, then* MIN-MAX KNAPSACK *can be solved in time* $O(nU^k)$.

**Corollary 9** MIN-MAX KNAPSACK *admits an fptas running in time* $O(\frac{n^{k+1}}{\varepsilon^k})$.

**Proof:** This results from Proposition 2 (for $f(n) = 1+\varepsilon$), Proposition 9, and Proposition 1. □

# 5 Conclusions

In this paper we have presented characterizations for the existence of an fptas for the min-max (regret) versions of several combinatorial optimization problems. These results lead to new fptas's with better running times than the ones previously presented in the literature. However, the applicability of these results is limited to problems where the min-max (regret) versions can be solved using a pseudo-polynomial algorithm (shortest path, spanning tree, knapsack, ...). Specialized techniques are thus needed for approximating strongly *NP*-hard min-max (regret) versions.

## References


[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullmann. *The design and analysis of computer algorithms*. Addison-Wesley, Reading, 1976.

[2] H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation of min-max and min-max regret versions of some combinatorial optimization problems. *European Journal of Operational Research*, 179(2):281–290, 2007. Preliminary version published in ESA 2005, Mallorca, LNCS 3669, 862-873.

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

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

[5] J. Edmonds. System of distinct representatives and linear algebra. *Journal of Research of the National Bureau of Standards*, 718(4):241–245, 1967.

[6] T. Erlebach, H. Kellerer, and U. Pferschy. Approximating multiobjective knapsack problems. *Management Science*, 48(12):1603–1612, 2002.

[7] R. Hassin. Approximation schemes for the restricted shortest path. *Mathematics of Operations Research*, 17(1):36–42, 1992.

[8] S. P. Hong, S. J. Chung, and B. H. Park. A fully polynomial bicriteria approximation scheme for the constrained spanning tree problem. *Operations Research Letters*, 32(3):233–239, 2004.

[9] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[10] P. W. Kasteleyn. Graph theory and theoretical physics. In *Graph theory and crystal physics*, pages 43–110. Academic Press London, 1967.

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

[12] M. Mahajan, P. R. Subramanya, and V. Vinay. The combinatorial approach yields an NC algorithm for computing Pfaffians. *Discrete Applied Mathematics*, 143(1-3):1–16, 2004.

[13] M. Mahajan and V. Vinay. Determinants: combinatorics, algorithms, and complexity. In *Proceedings of the Eigth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1997), New Orleans, USA*, pages 730–738, 1997.

[14] C. H. Papadimitriou. *Computational complexity*. Addison Wesley, 1994.

[15] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *IEEE Symposium on Foundations of Computer Science (FOCS 2000), Redondo Beach, California, USA*, pages 86–92, 2000.

[16] K. Pruhs and G. J. Woeginger. Approximation schemes for a class of subset selection problems. *Theoretical Computer Science*, 382(2):151–156, 2007.

[17] W.T. Tutte. *Graph Theory*, volume 21 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, 1984.

[18] A. Warburton. Approximation of Pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79, 1987.

[19] G. J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (fptas)? *INFORMS Journal on Computing*, 12(1):57–74, 2000.

[20] G. Yu. On the max-min 0-1 knapsack problem with robust optimization applications. *Operations Research*, 44(2):407–415, 1996.

# A Matrix tree theorem and Pfaffian orientations

We briefly recall classical results concerning the *matrix tree theorem* and *Pfaffian orientations* that enable us to derive approximation schemes for min-max and min-max regret versions of spanning tree and weighted perfect matching in planar graphs in sections 4.2 and 4.3 respectively.

The matrix tree theorem provides a way of counting all the spanning trees in a graph (see, e.g., [17]). Consider a graph $G = (V, E)$ with $|V| = n$, $|E| = m$ and let $c_{ij}$ denote the cost of edge $(i, j) \in E$.

Define an $n \times n$ matrix $A$ whose entries are given as follows:

$$a_{ij} = \begin{cases} -c_{ij} \text{ if } i \neq j \text{ and } (i,j) \in E \\ \sum_{(i,\ell)\in E} c_{i\ell} \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}$$

Define $A_r$ as the submatrix of $A$ obtained by deleting the $r^{th}$ row and $r^{th}$ column and $\mathcal{D}(A_r)$ as its determinant. The matrix tree theorem states that, for any $r \in \{1, \ldots, n\}$, the following equality holds:

$$\mathcal{D}(A_r) = \sum_{T\in\mathcal{T}} \prod_{(i,j)\in T} c_{ij} \tag{1}$$

where $\mathcal{T}$ is the set of all spanning trees of $G$.

As indicated in [4], this theorem can be extended to count the number of spanning trees of value $v$ for each possible value $v$ using a matrix depending on one variable. Following this idea, we can extend the matrix tree theorem to the multiple scenarios case as in [8]. Define the $n \times n$ matrix $A(y_1, \ldots, y_k)$ whose entries are given as follows:

$$a_{ij}(y_1, \ldots, y_k) = \begin{cases} -\prod_{s=1}^{k} y_s^{c_{ij}^s} \text{ if } i \neq j \text{ and } (i,j) \in E \\ \sum_{(i,\ell)\in E} \prod_{s=1}^{k} y_s^{c_{i\ell}^s} \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}$$

Then, the determinant of the submatrix $A_r(y_1, \ldots, y_k)$ obtained by deleting any $r^{th}$ row and $r^{th}$ column is given by

$$\mathcal{D}(A_r(y_1, \ldots, y_k)) = \sum_{v_1,\ldots,v_k\in V^{\mathcal{T}}} a_{v_1,\ldots,v_k} \prod_{s=1}^{k} y_s^{v_s} \tag{2}$$

where $a_{v_1,\ldots,v_k}$ is the number of spanning trees with value $v_s$ in scenario $s$, for all $s \in S$, and $V^{\mathcal{T}}$ is the set of values reached on all scenarios, for all spanning trees of $G$.

Equality (2) is obtained by replacing each $c_{ij}$ in (1) by $\prod_{s=1}^{k} y_s^{c_{ij}^s}$. Then each product term in (1) corresponding to tree $T$ becomes $\prod_{s=1}^{k} y_s^{\sum_{(i,j)\in T} c_{ij}^s}$.

Kasteleyn [10] gives an efficient procedure to count all perfect matchings in planar graphs. Let $G = (V, E)$ be a planar graph with $|V| = 2n$, and let $c_{ij}$ denote the weight of edge $(i, j) \in E$. Given an orientation of the edges, let $B$ denote a $2n \times 2n$ matrix defined as follows:

$$b_{ij} = \begin{cases} c_{ij} & \text{if } (i,j) \in E \text{ and } (i,j) \text{ is oriented from } i \text{ to } j \\ -c_{ij} & \text{if } (i,j) \in E \text{ and } (i,j) \text{ is oriented from } j \text{ to } i \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathcal{P}f(B)$ denote the Pfaffian of matrix $B$. Kasteleyn [10] gives an efficient algorithm to obtain an orientation of the graph such that the following identity holds:

$$\mathcal{P}f(B) = \sum_{M \in \mathcal{M}} \prod_{(i,j)\in M} c_{ij} \tag{3}$$

where $\mathcal{M}$ is the set of all matchings of $G$.

As for the matrix tree theorem, identity (3) can be extended to count the number of perfect matchings of value $v_1, \ldots, v_k$ for each possible profile of values, using a matrix depending on $k$ variables. Given an orientation of the edges, let $B(y_1, \ldots, y_k)$ denote a $2n \times 2n$ matrix defined as follows:

$$b_{ij}(y_1, \ldots, y_k) = \begin{cases} \prod_{s=1}^{k} y_s^{c_{ij}^s} & \text{if } (i,j) \in E \text{ and } (i,j) \text{ is oriented from } i \text{ to } j \\ -\prod_{s=1}^{k} y_s^{c_{ij}^s} & \text{if } (i,j) \in E \text{ and } (i,j) \text{ is oriented from } j \text{ to } i \\ 0 & \text{otherwise} \end{cases}$$

Following this extension we can adapt equality (3), in the same way as for the matrix tree theorem, obtaining:

$$\mathcal{P}f(B(y_1, \ldots, y_k)) = \sum_{v_1,\ldots,v_k \in V^{\mathcal{M}}} a_{v_1,\ldots,v_k} \prod_{s=1}^{k} y_s^{v_s} \tag{4}$$

where $a_{v_1,\ldots,v_k}$ is the number of matchings with value $v_s$ reached on scenario $s$, for all $s \in S$ and $V^{\mathcal{M}}$ is the set of values reached on all scenarios for all matchings of $G$.

# B Proof of Theorem 3

We prove here that min-max and min-max regret versions of Minimum Weighted Perfect Matching in planar graphs are *NP*-hard even for two scenarios. For this purpose, we use a reduction from a variant of the Partition problem, proved *NP*-hard [9], and defined as follows.

Even Odd Partition
**Input:** A finite set of positive integers $A = \{a_1, a_2, \ldots, a_{2n-1}, a_{2n}\}$.
**Question:** Is there a subset $A' \subseteq A$, containing exactly one of $a_{2i-1}, a_{2i}$ for $1 \leq i \leq n$, such

that $\sum_{a_p \in A'} a_p = \sum_{a_p \in A \setminus A'} a_p$?

*Proof of Theorem 3.* In order to obtain our result for Min-Max Weighted Perfect Matching in planar graphs, we construct a polynomial reduction from Even Odd Partition. Let $I$ be an instance of this problem on $2n$ integers $a_1, a_2, \ldots, a_{2n-1}, a_{2n}$. We construct an instance $G = (V, E)$ of Min-Max Weighted Perfect Matching with two scenarios $s_1, s_2$, such that $G$ is planar. The vertex set is $V = \{1, \ldots, 4n\}$, where vertices $2i-1, 2i, 2n+i$, and $3n+i$ correspond to integers $a_{2i-1}, a_{2i}$, for $i = 1, \ldots, n$. The edge set is $E = \{(2i-1, 2n+i), (2i-1, 3n+i), (2i, 2n+i), (2i, 3n+i) : i = 1, \ldots, n\}$. The edge weights for scenario $s_1$ and $s_2$ are defined as follows: $c^{s_1}_{2i-1,2n+i} = a_{2i-1}$, $c^{s_1}_{2i,2n+i} = a_{2i}$, $c^{s_2}_{2i-1,2n+i} = a_{2i}$ and $c^{s_2}_{2i,2n+i} = a_{2i-1}$ for $i = 1, \ldots, n$, and $c^s_{i,j} = 0$ for any other edge $(i, j)$ of $G$ and scenario $s$ (see Figure 1).

We show in the following that there exists a subset $A' \subseteq A$, containing exactly one of $a_{2i-1}, a_{2i}$ for $1 \leq i \leq n$, such that $\sum_{a_p \in A'} a_p = \sum_{a_p \in A \setminus A'} a_p$ if and only if there exists a perfect matching $M$ in $G$ such that $\max\{val(M, s_1), val(M, s_2)\} \leq \frac{1}{2} \sum_{a_p \in A} a_p$.

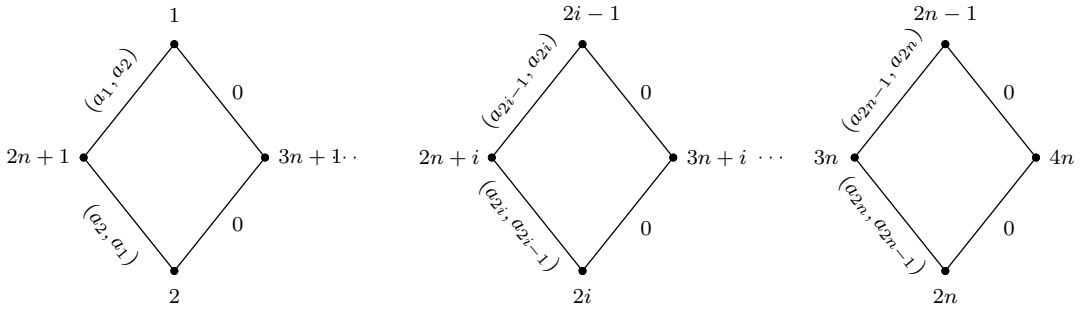


Figure 1: Min-Max Weighted Perfect Matching instance resulting from Even Odd Partition instance.

Suppose first that such a subset $A'$ exists. Consider the following matching $M$: if $a_{2i-1} \in A'$ then $M$ contains $(2i-1, 2n+i), (2i, 3n+i)$, and if $a_{2i} \in A'$ then $M$ contains $(2i-1, 3n+i), (2i, 2n+i)$. The values of $M$ in scenarios $s_1$ and $s_2$ are $val(M, s_1) = \sum_{a_p \in A'} a_p$ and $val(M, s_2) = \sum_{a_p \in A \setminus A'} a_p$. Since $\sum_{a_p \in A'} a_p = \sum_{a_p \in A \setminus A'} a_p$, we have $\max\{val(M, s_1), val(M, s_2)\} = \frac{1}{2} \sum_{a_p \in A} a_p$.

Suppose now that there exists a perfect matching $M$ in $G$ such that its value $\max\{val(M, s_1), val(M, s_2)\} \leq \frac{1}{2} \sum_{a_p \in A} a_p$. We consider the set $A'$ among the $2n$ integers defined as follows: if $M$ contains $(2i-1, 2n+i), (2i, 3n+i)$, then we introduce $a_{2i-1}$ in $A'$ and if $M$ contains $(2i-1, 3n+i), (2i, 2n+i)$ then we introduce $a_{2i}$ in $A'$. The values of $M$ in scenarios $s_1$ and $s_2$ are $val(M, s_1) = \sum_{a_p \in A'} a_p$ and $val(M, s_2) = \sum_{a_p \in A \setminus A'} a_p$. Since $\max\{val(M, s_1), val(M, s_2)\} \leq \frac{1}{2} \sum_{a_p \in A} a_p$, we have $\sum_{a_p \in A'} a_p = \sum_{a_p \in A \setminus A'} a_p$.

In order to obtain our result for Min-Max Regret Weighted Perfect Matching in planar graphs, we construct a polynomial reduction from Even Odd Partition. Let $I$ be an instance of this problem on $2n$ integers $a_1, a_2, \ldots, a_{2n-1}, a_{2n}$. We construct an instance $G = (V, E)$ of Min-Max Weighted Perfect Matching, with two scenarios $s_1, s_2$, such that $G$ is planar. The vertex set is $V = \{1, \ldots, 6n\}$, where vertices $2i-1, 2i, 2n+i, 3n+i, 4n+i$ and $5n+i$ correspond to integers $a_{2i-1}, a_{2i}$, for $i = 1, \ldots, n$. The edge set is $E = E_1 \cup E_2 \cup E_3$

where $E_1 = \{(2n+i, 3n+i), (4n+i, 5n+i) : i = 1, \ldots, n\}$, $E_2 = \{(2n+i, 5n+i), (3n+i, 4n+i) : i = 1, \ldots, n\}$ and $E_3 = \{(2i-1, 2n+i), (2i, 2n+i), (2i-1, 4n+i), (2i, 4n+i), (2i-1, 2i) : i = 1, \ldots, n\} \cup \{(3n+i, 5n+i+1) : i = 1, \ldots, n-1\} \cup \{(5n+1, 4n)\}$. The edge costs for scenarios $s_1$ and $s_2$ are defined as follows: $c^{s_1}_{i,j} = \sum_{a_p \in A} a_p$, for any edge $(i,j) \in E_1$, $c^{s_2}_{i,j} = \sum_{a_p \in A} a_p$, for any edge $(i,j) \in E_2$, $c^{s_1}_{2i-1,4n+i} = a_{2i-1}$, $c^{s_2}_{2i-1,4n+i} = a_{2i}$, $c^{s_1}_{2i,4n+i} = a_{2i}$, $v^{s_2}_{2i,4n+i} = a_{2i-1}$, for $i = 1, \ldots, n$, and $c^s_{i,j} = 0$, for any other edge $(i,j)$ and scenario $s \in S$ (see Figure 2).

Notice that the minimum weighted perfect matching $M^*_i$ in scenario $s_i$ verifies $val(M^*_i, s_i) = 0$, for $i = 1, 2$. Indeed, $M^*_1$ includes edges $(2n+i, 5n+i), (3n+i, 4n+i)$ and $(2i-1, 2i)$ for $i = 1, \ldots, n$. On the other hand, $M^*_2$ includes edges $(2n+i, 3n+i), (4n+i, 5n+i)$ and $(2i-1, 2i)$ for $i = 1, \ldots, n$.

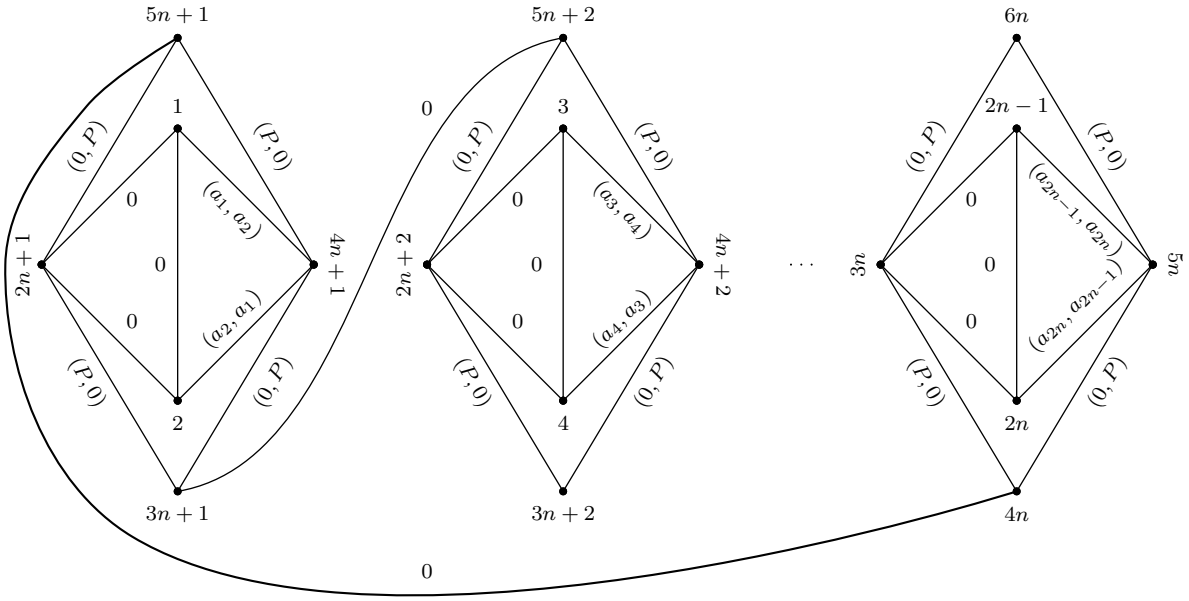


Figure 2: Min-Max Regret Weighted Perfect Matching instance resulting from Even Odd Partition instance.

We claim that there exists a subset $A' \subseteq A$, containing exactly one of $a_{2i-1}, a_{2i}$ for $1 \leq i \leq n$, such that $\sum_{a_p \in A'} a_p = \sum_{a_p \in A \setminus A'} a_p$ if and only if there exists a perfect matching $M$ in $G$ such that $R_{max}(M) \leq \frac{1}{2} \sum_{a_p \in A} a_p$.

Consider a subset $A' \subseteq A$, containing exactly one of $a_{2i-1}, a_{2i}$ for $1 \leq i \leq n$, such that $\sum_{a_p \in A'} a_p = \sum_{a_p \in A \setminus A'} a_p$. We construct a perfect matching $M$ in $G$ associated with $A'$. Matching $M$ contains $(2i-1, 4n+i)$ and $(2i, 2n+i)$ if $a_{2i-1} \in A'$ and $(2i, 4n+i)$ and $(2i-1, 2n+i)$ if $a_{2i} \in A'$. Moreover, $M$ contains edges $(3n+i, 5n+i+1)$ for $i = 1, \ldots, n-1$ and $(5n+1, 4n)$. Thus, we have $val(M, s_1) = \sum_{a_p \in A'} a_p$ and $val(M, s_2) = \sum_{a_p \in A \setminus A'} a_p$, which implies that $R_{max}(M) = \frac{1}{2} \sum_{a_p \in A} a_p$.

Conversely, consider a perfect matching $M$ in $G$ with $R_{max}(M) \leq \frac{1}{2} \sum_{a_p \in A} a_p$. Matching $M$ cannot contain one of the following edges $(2n+i, 3n+i)$, $(3n+i, 4n+i)$, $(4n+i, 5n+i)$ and $(2n+i, 5n+i)$ for some $i = 1, \ldots, n$, since otherwise $R_{max}(M) \geq \sum_{a_p \in A} a_p$. Thus, $M$ contains for $i = 1, \ldots, n$ either edges $(2i-1, 4n+i), (2i, 2n+i)$ or edges $(2i, 4n+i), (2i-1, 2n+i)$.

Moreover, $M$ contains edges $(3n+i, 5n+i+1)$ for $i = 1, \ldots, n-1$ and $(5n+1, 4n)$. We define from $M$ a subset $A' \subseteq A$ as follows : for $i = 1, \ldots, n$, $A'$ contains $a_{2i-1}$ if $(2i-1, 4n+i) \in M$ and $a_{2i}$ if $(2i, 4n+i) \in M$. Thus we have $R_{max}(M) = \max\{\sum_{a_p \in A'} a_p, \sum_{a \in A \setminus A'} a_p\}$ and since $R_{max}(M) \leq \frac{1}{2} \sum_{a_p \in A} a_p$, we have $\sum_{a_p \in A'} a_p = \sum_{a \in A \setminus A'} a_p$. □