Approximation complexity of min-max (regret) versions of shortest path, spanning tree, and knapsack

Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten

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

Abstract. This paper investigates, for the first time in the literature, the approximation of min-max (regret) versions of classical problems like shortest path, minimum spanning tree, and knapsack. For a bounded number of scenarios, we establish fully polynomial-time approximation schemes for the min-max versions of these problems, using relationships between multi-objective and min-max optimization. Using dynamic programming and classical trimming techniques, we construct a fully polynomial-time approximation scheme for min-max regret shortest path. We also establish a fully polynomial-time approximation scheme for min-max regret spanning tree and prove that min-max regret knapsack is not at all approximable. We also investigate the case of an unbounded number of scenarios, for which min-max and min-max regret versions of polynomial-time solvable problems usually become strongly *NP*-hard. In this setting, non-approximability results are provided for min-max (regret) versions of shortest path and spanning tree.

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

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. 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 and an upper bound. In the *discrete scenario case*, the scenario set is described explicitly. In this case, that is considered in this paper, we distinguish situations where the number of scenarios is bounded by a constant

* 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.

from those where the number of scenarios is unbounded. Kouvelis and Yu [5] 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 a good 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 of the value of the solution from the optimal value of the corresponding scenario.

Complexity of the min-max and min-max regret versions has been studied extensively during the last decade. In [5], for the discrete scenario case, the complexity of min-max and min-max regret versions of several combinatorial optimization problems was studied, including shortest path, minimum spanning tree, assignment, and knapsack problems. In general, these versions are shown to be harder than the classical versions. More precisely, if the number of scenarios is unbounded, 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 [5] to prove *NP*-difficulty are based on transformations from the partition problem which is known to be weakly *NP*-hard [3]. These reductions give no indications 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 (shortest path, knapsack, minimum spanning tree on grid graphs, ...). All these pseudo-polynomial algorithms described in [5] are based on dynamic programming.

In this paper we consider, for the first time in the literature, the approximation complexity of these versions for classical combinatorial optimization problems, focusing on three typical problems: shortest path, minimum spanning tree and knapsack.

After presenting preliminary concepts in Section 2, we investigate the existence of approximation algorithms for our reference problems when the number of scenarios is bounded by a constant (Section 3), and when it is unbounded (Section 4). The results we obtained are summarized in Table 1.

	bounded		unbounded	
	min-max	min-max regret	min-max	min-max regret
shortest path	fptas	fptas	not $(2-\varepsilon)$ approx.	not $(2-\varepsilon)$ approx.
min spanning tree	fptas	fptas	not $(\frac{3}{2}-\varepsilon)$ approx.	not $(\frac{3}{2}-\varepsilon)$ approx.
knapsack	fptas	not at all approx.	not at all approx.	not at all approx.

Table 1. Approximation results for min-max and min-max versions

2 Preliminaries

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

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

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, ...).

2.1 Min-max, min-max regret versions

Given a problem $\mathcal{P} \in \mathcal{C}$, the min-max (regret) version associated to $\mathcal{P}$ has as input a finite set of scenarios S where each scenario $s \in S$ is represented by a vector $(c_1^s, \ldots, c_n^s)$. We denote by $val(x,s) = \sum_{i=1}^{n} c_i^s x_i$ 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.

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 $opt(I)$ the optimum value of instance I. The *performance ratio* of x is $r(x) = \max\left\{\frac{val(x)}{opt(I)}, \frac{opt(I)}{val(x)}\right\}$, and its *error* is $\varepsilon(x) = r(x) - 1$.

For a function f, an algorithm is an *$f(n)$-approximation algorithm* if, for any instance I of the problem, it returns a solution x such that $r(x) \leq f(|I|)$. 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 set of problems having an fptas is denoted by *FPTAS*.

We recall the notion of *gap-introducing reduction* (see, e.g., [1, 8]). Let $\mathcal{P}$ be a decision problem and $\mathcal{Q}$ a minimization problem. $\mathcal{P}$ is gap-introducing reducible to $\mathcal{Q}$ if there exist two functions f and α such that, given an instance I of $\mathcal{P}$, it is possible to construct in polynomial time an instance I' of $\mathcal{Q}$, such that

- if I is a positive instance then $opt(I') \leq f(I')$,
- if I is a negative instance then $opt(I') > \alpha(|I'|)f(I')$.

If $\mathcal{P}$ is an *NP*-hard problem, and $\mathcal{P}$ is gap-introducing reducible to $\mathcal{Q}$, then $\mathcal{Q}$ is not $\alpha(n)$-approximable if $P \neq NP$.

2.3 Multi-objective optimization

It is natural to consider scenarios as criteria (or objective functions) and to investigate relationships between min-max (regret) and multi-objective optimization, when it is usually assumed that the number of criteria is a constant.

The multi-objective version associated to $\mathcal{P} \in \mathcal{C}$, denoted by Multi-objective $\mathcal{P}$, has for input k objective functions (or criteria) where the hth objective function has coefficients $c_1^h, \ldots, c_n^h$. We denote by $val(x,h) = \sum_{i=1}^{n} c_i^h x_i$ the value of solution $x \in X$ on criterion h, and assume w.l.o.g. that all criteria are to be minimized. Given two feasible solutions x and y, we say that y dominates x if $val(y,h) \leq val(x,h)$ for $h = 1, \ldots, k$ with at least one strict inequality. The problem consists of finding the set E of efficient solutions. A feasible solution x is *efficient* if there is no other feasible solution y that dominates x. In general Multi-objective $\mathcal{P}$ is intractable in the sense that it admits instances for which the size of E is exponential in the size of the input. A set F of feasible solutions is called an $f(n)$-approximation of the set of efficient solutions if, for every efficient solution x, F contains a feasible solution y such that $val(y,h) \leq f(n)val(x,h)$ for each criterion $h = 1, \ldots, k$. An algorithm is an *$f(n)$-approximation algorithm* for a multi-objective problem, if for any instance I of the problem it returns an $f(n)$-approximation of the set of efficient solutions. A multi-objective problem has an *fptas* if, for every constant $\varepsilon > 0$, there exists an $(1+\varepsilon)$-approximation algorithm for the set of efficient solutions which is polynomial both in the size of the input and in $1/\varepsilon$.

3 Bounded number of scenarios

3.1 Min-max problems

Consider a minimization problem $\mathcal{P}$. It is easy to see that at least one optimal solution for Min-Max $\mathcal{P}$ is necessarily an efficient solution. Indeed, if $x \in X$ dominates $y \in X$ then $\max_{s \in S} val(x,s) \leq \max_{s \in S} val(y,s)$. Therefore, we obtain an optimal solution for Min-Max $\mathcal{P}$ by taking, among the efficient solutions, one that has a minimum $\max_{s \in S} val(x,s)$. Observe, however, that if Min-Max $\mathcal{P}$ admits several optimal solutions, some of them may not be efficient, but at least one is efficient.

Theorem 1. *For any function $f : \mathbb{N} \to (1, \infty)$, if* Multi-objective $\mathcal{P}$ *has a polynomial-time $f(n)$-approximation algorithm, then* Min-Max $\mathcal{P}$ *has a polynomial-time $f(n)$-approximation algorithm.*

Proof. Let F be an $f(n)$-approximation of the set of efficient solutions. Since at least one optimal solution x^* for Min-Max $\mathcal{P}$ is efficient, there exists a solution $y \in F$ such that $val(y,s) \leq f(n)val(x^*,s)$, for $s \in S$. Consider among the set F a solution z that has a minimum $\max_{s \in S} val(z,s)$. Thus, $\max_{s \in S} val(z,s) \leq \max_{s \in S} val(y,s) \leq \max_{s \in S} f(n)val(x^*,s) = f(n)opt(I)$. □

Corollary 1. *For a bounded number of scenarios,* Min-Max Shortest Path*,* Min-Max Spanning Tree*, and* Max-Min Knapsack *are in FPTAS.*

Proof. For a bounded number of criteria, multi-objective versions of Shortest Path, Minimum Spanning Tree and Knapsack, have an fptas [6]. □

3.2 Min-max regret problems

General results

As for min-max, at least one optimal solution for Min-Max Regret $\mathcal{P}$ is necessarily an efficient solution for Multi-objective $\mathcal{P}$. Indeed, if $x \in X$ dominates $y \in X$ then $val(x,s) \leq val(y,s)$, for each $s \in S$, and thus $R_{max}(x) \leq R_{max}(y)$. Therefore, we obtain an optimal solution for Min-Max Regret $\mathcal{P}$ by taking, among the efficient solutions, a solution x that has a minimum $R_{max}(x)$. Unfortunately, given F an $f(n)$-approximation of the set of efficient solutions, a solution $x \in F$ with a minimum $R_{max}(x)$ is not necessarily an $f(n)$-approximation for the optimum value since the minimum maximum regret could be very small compared with the error that was allowed in F.

The following result deals with problems whose feasible solutions have a fixed size. In this context, we need to consider instances where some coefficients are negative but any feasible solution has a non-negative value. For an optimization problem $\mathcal{P}$, we denote by $\mathcal{P}'$ the extension of $\mathcal{P}$ to these instances.

Theorem 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 $\mathcal{P}'$ *has a polynomial-time $f(n)$-approximation algorithm, then* Min-Max Regret $\mathcal{P}$ *has a polynomial-time $f(n)$-approximation algorithm.*

Proof. Let t be the size of all feasible solutions of any instance of $\mathcal{P}$. Consider an instance I of Min-Max Regret $\mathcal{P}$ where c_i^s is the value of coefficient c_i in scenario $s \in S$. Compute for each scenario s the value val_s^* of an optimum solution. We construct from I an instance $\overline{I}$ of Min-Max $\mathcal{P}'$ with the same number of scenarios, where $\overline{c_i^s} = c_i^s - \frac{val_s^*}{t}$. Remark that some coefficients could be negative but any feasible solution has a non-negative value. Let $\overline{val}(x,s)$ denote the value of solution x in scenario s in $\overline{I}$. The sets of the feasible solutions of both instances are the same and moreover, for any feasible solution x, and for any scenario $s \in S$, we have $R(x,s) = val(x,s) - val_s^* = \overline{val}(x,s)$ since any feasible solution is of size t. Therefore, an optimum solution for I is also an optimum solution for $\overline{I}$ with $opt(\overline{I}) = opt(I)$. □

Min-Max Regret Spanning Tree

Corollary 2. Min-Max Regret Spanning Tree*, with a bounded number of scenarios, is in FPTAS.*

Proof. By combining the rounding technique [7], with a pseudo-polynomial time algorithm for the exact version of minimum spanning tree [2, 4], that can be extended to instances with negative coefficients, we can obtain an fptas for Min-Max Regret (Spanning Tree)$'$. Using Theorem 2, the result follows. □

Min-Max Regret Shortest Path

We construct in the following an fptas for Min-Max Regret Shortest Path considering the multi-objective problem that consists of enumerating the paths whose regret vectors are efficient.

Theorem 3. Min-Max Regret Shortest Path*, with a bounded number of scenarios, is in FPTAS.*

Proof. We consider first the case when the graph is acyclic and we describe briefly at the end how to adapt this procedure for graphs with cycles.

Consider an instance I described by a directed acyclic graph $G = (V, A)$, where $V = \{1, \ldots, n\}$ is such that if $(i, j) \in A$ then $i < j$, and a set S of k scenarios describing for each arc $(i, j) \in A$ its cost in scenario s by c_{ij}^s. Denote by c_{ij} the vector of size k formed by c_{ij}^s, $s \in S$. Let $(val_s^*)^i$, $s \in S$, $1 \leq i \leq n$ be the value of a shortest path in graph G from 1 to i under scenario s and let $(val^*)^i$ be the vector of size k of these values $(val_s^*)^i$, $s \in S$.

In the following, we describe firstly a dynamic programming algorithm that computes at each stage i, $1 \leq i \leq n$, the set R^i of efficient vectors of regrets for paths from 1 to i, for each scenario $s \in S$. 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 $r_s^j = val(P_i, s) + c_{ij}^s - (val_s^*)^j = r_s^i + (val_s^*)^i + c_{ij}^s - (val_s^*)^j$, $s \in S$. The algorithm starts by initializing $R^1 = \{(0, \ldots, 0)\}$, where $(0, \ldots, 0)$ is a vector of size k and for $2 \leq j \leq n$ let

$$R^j = \text{Min}_{i \in \Gamma^{-1}(j)}\{r^i + (val^*)^i + c_{ij} - (val^*)^j : r^i \in R^i\}$$

where the operator "Min" preserves the efficient vectors.

Observe that, for $2 \leq j \leq n$, R^j, which contains all efficient regret vectors for paths from 1 to j, necessarily contains one optimal vector corresponding to a min-max regret shortest path from 1 to j. We also point out that, for this algorithm as well as for the following approximation algorithm, any path of interest can be obtained using standard bookkeeping techniques that do not affect the complexity of these algorithms.

Our approximation algorithm is a dynamic programming procedure combined with a trimming of the states depending on an accepted error $\varepsilon > 0$. In this procedure, define set $T^1 = \{(0, \ldots, 0)\}$, and sets U^j, T^j for $2 \leq j \leq n$ as follows

$U^j = \cup_{i \in \Gamma^{-1}(j)}\{r^i + (val^*)^i + c_{ij} - (val^*)^j : r^i \in T^i\}$,
$T^j = Red(U^j)$, where Red is an operator satisfying the following property

$$\forall\, r \in U^j, \exists\, \bar{r} \in T^j : \bar{r} \leq r(1+\varepsilon)^{\frac{1}{n-1}}$$

where, given two vectors r', r'' of size $|S|$, we have $r' \leq r''$ if and only if $r'_s \leq r''_s$, $\forall s \in S$.

In the following, we prove by induction on j the proposition

$$P(j) : \forall\, r \in R^j, \exists\, \tilde{r} \in T^j \text{ such that } \tilde{r} \leq r(1+\varepsilon)^{\frac{j-1}{n-1}}$$

Obviously, proposition $P(1)$ is true. Supposing now that $P(i)$ is true for $i < j$, we show that $P(j)$ is true. Consider $r \in R^j$. Then there exists $i < j$ such that $(i,j) \in A$ and $r' \in R^i$ such that $r = r' + (val^*)^i + c_{ij} - (val^*)^j$. Since $(val^*)^i + c_{ij} \geq (val^*)^j$, we have $r \geq r'$. Using the induction hypothesis for i, there exists $\tilde{r} \in T^i$ such that $\tilde{r} \leq r'(1+\varepsilon)^{\frac{i-1}{n-1}}$. Since $\tilde{r} \in T^i$ and $(i,j) \in A$, we have $\tilde{r} + (val^*)^i + c_{ij} - (val^*)^j \in U^j$ and, using the property satisfied by *Red*, there exists $\bar{r} \in T^j$ such that:

$$\bar{r} \leq [\tilde{r} + (val^*)^i + c_{ij} - (val^*)^j](1+\varepsilon)^{\frac{1}{n-1}} \leq$$
$$\leq [r'(1+\varepsilon)^{\frac{i-1}{n-1}} + r - r'](1+\varepsilon)^{\frac{1}{n-1}} \leq r(1+\varepsilon)^{\frac{i}{n-1}} \leq r(1+\varepsilon)^{\frac{j-1}{n-1}}.$$

Thus proposition $P(j)$ is true for $j = 1, \ldots, n$. Obviously, there exists $r \in R^n$ such that $opt(I) = \max_{s \in S} r_s$. Applying $P(n)$ to $r \in R^n$, there exists $\tilde{r} \in T^n$ such that $\tilde{r} \leq r(1+\varepsilon)$ and thus $\max_{s \in S} \tilde{r_s} \leq (1+\varepsilon) opt(I)$.

We show in the following how this algorithm can be implemented in polynomial time in $|I|$ and $\frac{1}{\varepsilon}$. Let $c_{max} = \max_{(i,j) \in A, s \in S} c_{ij}^s$. For any $s \in S$ and $2 \leq j \leq n$, we have $r_s^j \leq (n-1)c_{max}$. An operator *Red* can be implemented in polynomial time using the technique of interval partitioning described in Sahni [7]. The idea is to partition the domain of values, for each scenario, into subintervals such that the ratio of the extremities is $(1+\varepsilon)^{\frac{1}{n-1}}$. Thus on each coordinate (or scenario) we have $\lceil \frac{(n-1)\log(n-1)c_{max}}{\log(1+\varepsilon)} \rceil$ subintervals. Operator *Red* can be implemented by selecting only one vector in each non-empty hypercube of the cartesian product of subintervals. Thus $|T^j| \leq (\frac{n \log n c_{max}}{\log(1+\varepsilon)})^k$, $2 \leq j \leq n$ and the time complexity of our algorithm is $O(n(\frac{n \log n c_{max}}{\log(1+\varepsilon)})^k)$ that is polynomial in $|I| = |A| k \log c_{max}$ and $\frac{1}{\varepsilon}$.

Consider now graphs with cycles. We can generalize the previous procedure, by defining a dynamic programming scheme with stages ℓ, $\ell = 1, \ldots, n-1$, containing sets of states R_j^ℓ which represent the set of efficient vectors of regrets for paths from 1 to j of length at most ℓ, $j = 2, \ldots, n$. □

Min-Max Regret Knapsack

In this section, we prove that Min-Max Regret Knapsack is not at all approximable even for two scenarios. For this, we use a reduction from Maximum Constrained Partition defined in [1].

Maximum Constrained Partition
Input: A finite set A and an integer size $s(a)$ for each $a \in A$, one element $a_0 \in A$ and a subset $B \subseteq A$.
Output: A feasible partition, i.e., a partition $(A', A \setminus A')$, $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$, with a maximum number of elements from B on the same side of the partition as a_0.

Maximum Constrained Partition was proved not approximable within $|A|^\varepsilon$ for some $\varepsilon > 0$ [9], but even deciding the existence of a feasible partition is *NP*-hard [3].

Theorem 4. *For any function $f : \mathbb{N} \to (1, \infty)$,* Min-Max Regret Knapsack *is not $f(n)$-approximable even for two scenarios, unless $P = NP$.*

Proof. We construct a reduction from Maximum Constrained Partition to Min-Max Regret Knapsack. Consider an instance I of Maximum Constrained Partition characterized by a set $A = \{a_0, a_1, \ldots, a_{n-1}\}$, a size $s(a)$ for each $a \in A$, and a subset $B \subseteq A$. We define an instance I' of Min-Max Regret Knapsack as follows: the number of items is $n+1$, the knapsack capacity is $d = \frac{1}{2}\sum_{a \in A} s(a)$, the items weights are $w_i = s(a_i)$ for $i = 0, \ldots, n-1$ and $w_n = d$. I' contains two scenarios and the values of the n items are defined as follows: $v_0^1 = n^3 d$, $v_i^1 = 0$, for $i = 1, \ldots, n$ and $v_i^2 = n^2 s(a_i) + \delta_i$, for $i = 0, \ldots, n-1$, where $\delta_i = 1$ if $a_i \in B$ and $\delta_i = 0$ otherwise, and $v_n^2 = n^2 d$.

Clearly, the optimum value in the first scenario of I' is $n^3 d$. If I does not contain a feasible partition, the optimum value in the second scenario is $n^2 d$. Indeed, candidate optimal solutions are either item n only with value $n^2 d$ or subsets T of items such that $\sum_{i \in T} s(a_i) < d$ with value $n^2 \sum_{i \in T} s(a_i) + \sum_{i \in T} \delta_i \leq n^2(d-1) + n < n^2 d$. If I contains a feasible partition, let $opt(I)$ denote its optimal value and $(A', A \setminus A')$ an optimal partition. Suppose that $a_0 \in A'$, otherwise we exchange A' with $A \setminus A'$. In this case, the optimum value in the second scenario is $n^2 d + opt(I)$ (the optimal solution is formed by items corresponding to elements from A').

If a solution x of I' does not contain a_0 then $R_{max}(x) = n^3 d$. Consequently, any optimal solution x^* of I' must include item a_0. If there exists a feasible partition in I, then we have $opt(I') = R_{max}(x^*) = 0$ and $opt(I') \geq n^2 - n$, otherwise.

Hence, Min-Max Regret Knapsack is not approximable since otherwise, any polynomial-time approximation algorithm for this problem applied to I' could decide if I contains a feasible partition (we could even derive a maximum constrained partition by selecting for A' the set of items present in the optimal solution). □

We conclude this section giving some precisions about the complexity status of these problems. Pseudo-polynomial time algorithms were given in the case of a bounded number of scenarios for min-max (max-min) and min-max regret versions of shortest path, knapsack, and minimum spanning tree on grid graphs [5]. Our fptas for min-max and min-max regret spanning tree establish also the existence of pseudo-polynomial time algorithms for these problems. Thus min-max (max-min) and min-max regret versions of shortest path, minimum spanning tree and knapsack are weakly *NP*-hard.

4 Unbounded number of scenarios

When the number of scenarios is unbounded, min-max and min-max regret shortest path as well as min-max spanning tree and max-min knapsack were proved strongly *NP*-hard in [5]. We establish the strong *NP*-hardness of min-max regret knapsack and min-max regret spanning tree in Theorems 5 and 9 respectively.

Concerning approximability results, reductions used in [5] for proving the strong *NP*-hardness of min-max/min-max regret shortest path, and min-max spanning tree, which are based on the 3-partition problem, cannot be used to establish non-approximability results for these problems. Using alternative reductions, we establish such results in Theorems 6-9. On the other hand, the reduction used in [5] for proving the strong *NP*-hardness of max-min knapsack is stronger and can be used to establish non-approximability results. In fact, it is a gap-introducing reduction from the set covering problem which maps positive instances into instances with optimum value at least 1 and negative instances into instances with optimum value 0. Therefore, we can deduce from this reduction that Max-Min Knapsack is not $f(n)$-approximable for any function $f : \mathbb{N} \to (1, \infty)$. Finally, regarding Min-Max Regret Knapsack, we know already that it is not $f(n)$-approximable for any function $f : \mathbb{N} \to (1, \infty)$, since even for two scenarios it is not approximable as shown in Theorem 4.

Now we state and prove the above-mentioned results.

Theorem 5. Min-Max Regret Knapsack*, with an unbounded number of scenarios, is strongly NP-hard.*

Proof. We construct a gap-introducing reduction from Vertex Cover. Given a graph $G = (V, E)$ on n vertices and m edges and a positive integer k, we define an instance I of Min-Max Regret Knapsack with n items and a set of m scenarios $S = \{s_1, \ldots, s_m\}$. The weights are $w_i = 1$, for any $i = 1, \ldots, n$, the knapsack capacity is $d = k$ and the value of item i in scenario s_j is $v_i^{s_j} = 1$ if node $i \in V$ is incident to edge $j \in E$, and 0 otherwise.

Observe first that $val^*_{s_j} = 2$, for all $s_j \in S$, which is obtained by taking the two items corresponding to the extremities of edge j. If G has a vertex cover V' of size at most k then the subset of items x' corresponding to V' has $val(x', s_j) \geq 1$, for any $s_j \in S$ since edge j is covered by V'. Thus, $R_{max}(x') \leq 1$, which implies $opt(I) \leq 1$.

If G has no vertex cover of size at most k then for any $V' \subseteq V$, $|V'| \leq k$, there exists $s_j \in S$, corresponding to an edge j which is not covered by V', such that the subset of items x' corresponding to V' has $val(x', s_j) = 0$, and thus $R_{max}(x') = 2$, which implies $opt(I) = 2$.

The existence of a polynomial-time algorithm would allow us to decide for Vertex Cover in polynomial time. □

Observe that the $(2 - \varepsilon)$ non-approximability result that could be derived from this proof is weaker than the result stated in Theorem 4.

We show in the following a non-approximability result for min-max and min-max regret versions of shortest path. For this, we use a reduction from Path With Forbidden Pairs that is known to be *NP*-hard [3].

Path With Forbidden Pairs
Input: A directed graph $G = (V, A)$, where $V = \{1, \ldots, n\}$, a collection $C = \{(a_1, b_1), \ldots, (a_t, b_t)\}$ of arcs from A.

Question: Is there a path from 1 to n in G containing at most one vertex from each arc of C ?

Theorem 6. Min-Max Shortest Path*, with an unbounded number of scenarios, is not* $(2-\varepsilon)$*-approximable, for any* $\varepsilon > 0$*, unless* $P = NP$*.*

Proof. We construct a gap-introducing reduction from Path With Forbidden Pairs. Let I be an instance of this problem with n vertices and m arcs, and t arcs in collection C. We construct an instance I' of Min-Max Shortest Path as follows: consider the same graph $G = (V, A)$, a scenario set $S = \{s_1, \ldots, s_t\}$, and costs of arcs defined for each scenario as

$$c_{ij}^{s_h} = \begin{cases} 2 \text{ if arc } (i,j) \text{ corresponds to } (a_h, b_h) \\ 1 \text{ if } i = a_h \text{ or } j = b_h, (i,j) \neq (a_h, b_h) \\ 0 \text{ otherwise} \end{cases}$$

Suppose that I is a positive instance, that is G contains a path p from 1 to n that has at most one extremity from each of the t arcs of C. Then for any scenario s, we have $val(p, s) \leq 1$. Then $\max_{s \in S} val(p, s) \leq 1$, which implies $opt(I') \leq 1$.

If I is a negative instance, then every path p from 1 to n in G contains either an arc or both extremities of an arc (a_h, b_h) from C. Then $val(p, s_h) = 2$ in both cases. Thus $\max_{s \in S} val(p, s) = 2$, which implies $opt(I') = 2$. □

Theorem 7. Min-Max Regret Shortest Path*, with an unbounded number of scenarios, is not* $(2-\varepsilon)$*-approximable, for any* $\varepsilon > 0$*, unless* $P = NP$*.*

Proof. As for the previous theorem, we construct a similar gap-introducing reduction from Path With Forbidden Pairs. Let I be an instance of this problem with n vertices and m arcs, and t arcs in the collection C. We construct an instance I'' of Min-Max Regret Shortest Path as follows: consider graph $G' = (V', A')$, where $V' = V \cup \{n+1, \ldots, n+|S|\}$, $A' = A \cup \{(1,i) : i = n+1, \ldots, n+|S|\} \cup \{(i,n) : i = n+1, \ldots, n+|S|\}$, and a scenario set $S = \{s_1, \ldots, s_t\}$. The costs of arcs in A are defined for each scenario $s \in S$ as in the previous theorem, and for any $s \in S$

$$c^s_{1,n+i} = c^s_{n+i,n} = \begin{cases} 0 \text{ if } \ s = s_i \\ 1 \text{ if } s \neq s_i \end{cases}$$

Obviously, $val^*_{s_i} = 0$ since the path $(1,\ n+i,\ n)$ has value 0 on scenario s_i. As previously, we can prove that if I is a positive instance, then $opt(I'') \leq 1$, otherwise $opt(I'') = 2$. □

We show in the following non-approximability results for min-max and min-max regret versions of spanning tree. The first result uses a reduction from Minimum Degree Spanning Tree that is known to be not $(\frac{3}{2} - \varepsilon)$-approximable, for any $\varepsilon > 0$ [3].

Minimum Degree Spanning Tree
Input: A graph $G = (V, E)$.
Output: A spanning tree such that its maximum degree is minimum.

Theorem 8. MIN-MAX SPANNING TREE, *with an unbounded number of scenarios, is not* $(\frac{3}{2}-\varepsilon)$*-approximable, for any* $\varepsilon > 0$*, unless* $P = NP$.

Proof. We construct an approximation preserving reduction from MINIMUM DEGREE SPANNING TREE. Let $G = (V, E)$ be an instance of this problem on n vertices. We construct an instance of MIN-MAX SPANNING TREE on the same graph G, with a set of n scenarios $S = \{s_1, \ldots, s_n\}$, and costs of edges in scenario s_h defined by $c_{ij}^{s_h} = 1$ if $h = i$ or $h = j$ and 0, otherwise. Then for any spanning tree T of G, the degree of $i \in V$ in T is the same as $val(T, s_i)$. Thus, the maximum degree of T, that is $\max_{i \in V} d_T(i)$, coincides with the maximum value of T over all scenarios from S, that is $\max_{s \in S} val(T, s)$. □

Theorem 9. MIN-MAX REGRET SPANNING TREE, *with an unbounded number of scenarios, is strongly NP-hard. Moreover, it is not* $(\frac{3}{2}-\varepsilon)$*-approximable, for any* $\varepsilon > 0$*, unless* $P = NP$.

Proof. We construct a gap-introducing reduction from 3SAT. Given a set $U = \{u_1, \ldots, u_n\}$ of boolean variables and a formula ϕ containing the clauses $\{C_1, \ldots, C_m\}$ over U such that each clause depends on exactly 3 variables, we construct an instance I of MIN-MAX REGRET SPANNING TREE defined on a graph $G = (V, E)$ where $V = \{1, \ldots, n\} \cup \{\overline{1}, \ldots, \overline{n}\} \cup \{n+1, \ldots, 3n\}$. Vertices $i, \bar{i}$, correspond to variable u_i, $i = 1, \ldots, n$. Edge set is $E = \{(i, n+i), (i, 2n+i), (\bar{i}, n+i), (\bar{i}, 2n+i), (i, \bar{i}) : i = 1, \ldots, n\} \cup \{(\bar{i}, i+1) : i = 1, \ldots, n-1\}$. Scenario set $S = S_1 \cup S_2 \cup S_3$ where $S_1 = \{s_1, \ldots, s_m\}$ corresponds to clauses and $S_2 = \{s'_{n+1}, \ldots, s'_{3n}\}$, $S_3 = \{s'_1, \ldots, s'_n, s'_{\overline{1}}, \ldots, s'_{\overline{n}}\}$ correspond to vertices of G. The costs of edges in scenario $s_j \in S_1$ are defined as follows: $c_{i,2n+i}^{s_j} = 1$ if $u_i \in C_j$, $c_{\bar{i},2n+i}^{s_j} = 1$ if $\overline{u_i} \in C_j$, and 0 otherwise. The values of edges in scenario $s'_j \in S_2$ are defined as follows: $c_{i,n+i}^{s'_{n+i}} = c_{\bar{i},n+i}^{s'_{n+i}} = n$, $c_{i,2n+i}^{s'_{2n+i}} = c_{\bar{i},2n+i}^{s'_{2n+i}} = n$, for every $i = 1, \ldots, n$ and 0 otherwise. The values of edges in scenario $s'_j \in S_3$ are defined as follows: $c_{i,n+i}^{s'_i} = c_{i,2n+i}^{s'_i} = c_{i\bar{i}}^{s'_i} = 2$, $c_{\bar{i},n+i}^{s'_{\bar{i}}} = c_{\bar{i},2n+i}^{s'_{\bar{i}}} = c_{i\bar{i}}^{s'_{\bar{i}}} = 2$, for every $i = 1, \ldots, n$ and 0 otherwise.

We compute in the following the optimum costs corresponding to each scenario. For any scenario $s_j \in S_1$, consider the spanning tree containing $\{(\bar{i}, i+1) : i = 1, \ldots, n-1\}$ and $\{(i, n+i), (\bar{i}, 2n+i), (i, \bar{i})\}$, for every i such that $u_i \in C_j$, or $\{(i, 2n+i), (\bar{i}, n+i), (i, \bar{i})\}$, otherwise. Obviously, this tree has value 0 in scenario s_j. For any scenario $s'_{n+i} \in S_2$, $val^*_{s'_{n+i}} = n$ since any spanning tree contains one of the edges $(i, n+i), (\bar{i}, n+i)$. Similarly, $val^*_{s'_{2n+i}} = n$, for all $s'_{2n+i} \in S_2$. For any scenario $s'_i \in S_3$, $val^*_{s'_i} = 2$ since any spanning tree contains at least one of the edges $(i, n+i), (i, 2n+i), (i, \bar{i})$. Similarly, $val^*_{s'_{\bar{i}}} = 2$, for all $s'_{\bar{i}} \in S_3$.

A spanning tree in G necessarily contains edges $(\bar{i}, i+1)$, $i = 1, \ldots, n-1$. We show in the following that every spanning tree T that contains edges $(i, \bar{i}), (i, n+i), (\bar{i}, 2n+i)$ or edges $(i, \bar{i}), (i, 2n+i), (\bar{i}, n+i)$ for every $i = 1, \ldots, n$, has $R_{max}(T) \leq 3$. Moreover, any other spanning tree T' in G has $R_{max}(T') \geq 4$. We have $val(T, s_j) \leq 3$, for any $s_j \in S_1$, $val(T, s'_j) = n$, for any $s'_j \in S_2$, and

$val(T, s'_j) = 4$, for any $s'_j \in S_3$. Thus, $R_{max}(T) \leq 3$. If T' contains both edges $(i, n+i), (\bar{i}, n+i)$ for some i, then $val(T', s'_{n+i}) = 2n$ and thus $R_{max}(T') = n$. We can also see that if a spanning tree T' contains both edges $(i, 2n+i), (\bar{i}, 2n+i)$ for some i, then $val(T', s'_{2n+i}) = 2n$ and thus $R_{max}(T') = n$. Consider in the following spanning trees T' that contain edges $(i, \bar{i})$, $i = 1, \ldots, n$. If T' contains both edges $(i, n+i), (i, 2n+i)$ for some i, then $val(T', s'_i) = 6$ and thus $R_{max}(T') = 4$. We can see also that if T' contains both edges $(\bar{i}, n+i), (\bar{i}, 2n+i)$ for some i, then $val(T', s'_{\bar{i}}) = 6$ and thus $R_{max}(T') = 4$. Thus an optimum solution in G is a spanning tree T that contains edges $(\bar{i}, i+1)$, $i = 1, \ldots, n-1$, edges $(i, \bar{i})$, $i = 1, \ldots, n$, and, for every $i = 1, \ldots, n$, it contains either edges $(i, n+i), (\bar{i}, 2n+i)$ or edges $(i, 2n+i), (\bar{i}, n+i)$. Such spanning trees are in one-to-one correspondence with assignments of variables $u_1, \ldots, u_n$. More precisely, T contains for some i edges $(i, n+i), (\bar{i}, 2n+i)$ if and only if u_i takes value 1, and it contains edges $(i, 2n+i), (\bar{i}, n+i)$ if and only if u_i takes value 0. If ϕ is satisfiable, then there exists an assignment x for $u_1, \ldots, u_n$ that satisfies each clause. Then, consider the spanning tree T associated to x. Every clause C_j is satisfied by x. Therefore, there exists $u_i \in C_j$, such that u_i has value 1 in x or $\overline{u_i} \in C_j$, such that u_i has value 0 in x. In both cases, $val(T, s_j) \leq 2$, for any $s_j \in S_1$. Tree T has also $val(T, s) = n$, for any $s \in S_2$ and $val(T, s) = 4$, for any $s \in S_3$, and thus, $R_{max}(T) = 2$, which implies $opt(I) = 2$.

Suppose now that ϕ is not satisfiable, that is for any assignment x, there exists a clause C_j that is not satisfied. Therefore, for any spanning tree T associated to x, we have $val(T, s_j) = 3$, and thus $R_{max}(T) = 3$, which implies $opt(I) = 3$. □

References

1. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation. Combinatorial optimization problems and their approximability properties.* Springer, 1999.
2. F. Barahona and R. Pulleyblank. Exact arborescences, matching and cycles. *Discrete Applied Mathematics*, 16:91–99, 1987.
3. M. Garey and D. Johnson. *Computer and Intractability: A Guide to the theory of NP-completeness.* Freeman, 1979.
4. 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*, 32:233–239, 2004.
5. P. Kouvelis and G. Yu. *Robust Discrete Optimization and its Applications.* Kluwer Academic Publishers, Boston, 1997.
6. 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*, pages 86–92, 2000.
7. S. Sahni. General techniques for combinatorial approximation. *Operations Research*, 25(6):920–936, 1977.
8. V. V. Vazirani. *Approximation Algorithms.* Springer, 2001.
9. D. Zuckerman. NP-complete problems have a version that's hard to approximate. In *Proc. 8th Annual Conference on Structure in Complexity Theory*, pages 305–312, 1993.