Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 179 (2007) 281–290

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Discrete Optimization

Approximation of min–max and min–max regret versions of some combinatorial optimization problems

Hassene Aissi, Cristina Bazgan, Daniel Vanderpooten *

LAMSADE, Université Paris-Dauphine, Place du Marechal de Lattre de Ta., 75775 Paris Cedex 16, France

Received 13 October 2005; accepted 16 March 2006
Available online 3 May 2006

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 constant 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. For a non-constant number of scenarios, in which case min–max and min–max regret versions of polynomial-time solvable problems usually become strongly NP-hard, non-approximability results are provided for min–max (regret) versions of shortest path and spanning tree.
© 2006 Elsevier B.V. All rights reserved.

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 coefficients of the objective function, 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 we address in this paper, we distinguish situations where the number of scenarios is constant from those where the number of scenarios is non-constant. Kouvelis and Yu [8] 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 the maximum deviation, over all

* Corresponding author.
E-mail addresses: aissi@lamsade.dauphine.fr (H. Aissi), bazgan@lamsade.dauphine.fr (C. Bazgan), vdp@lamsade.dauphine.fr (D. Vanderpooten).

possible scenarios, 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. Kouvelis and Yu [8] established the complexity of min–max and min–max regret versions, for the discrete scenario case, of several combinatorial optimization problems, including shortest path, minimum spanning tree, assignment, and knapsack problems. In general, these versions are shown to be harder than the classical versions. Min–max (regret) versions of polynomial problems usually become weakly NP-hard for a constant number of scenarios, and strongly NP-hard for a non-constant number of scenarios.

In this paper we consider, 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 constant number of scenarios, we establish fully polynomial-time approximation schemes (fptas) for the min–max versions of these problems, using relationships between multi-objective and min–max optimization. The interest of studying these relationships is that fptas, which determine an approximation of the non-dominated set (or Pareto set), have been proposed for the multi-objective version (see, e.g., [3,11,12]). This allows us to derive the existence of fptas for min–max versions of our reference problems. Concerning min–max regret versions, relationships with multi-objective versions still apply but cannot be used to derive the existence of fptas. Using dynamic programming and classical trimming techniques, we construct an fptas for min–max regret shortest path. We also give an fptas for min–max regret spanning tree and prove that min–max regret knapsack is not at all approximable. For a non-constant number of scenarios, non-approximability results are provided for min–max (regret) versions of shortest path and spanning tree. All the results are summarized in Table 1.

After presenting preliminary concepts in Section 2, we investigate the existence of approximation algorithms for our reference problems when the number of scenarios is constant (Section 3), and when it is non-constant (Section 4).

2. Preliminaries

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

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

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-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, min–max regret versions

Given a problem $\mathscr{P} \in \mathscr{C}$, the min–max (regret) version associated to $\mathscr{P}$ has as input a finite set of scenarios S where each scenario $s \in S$ is represented by a vector $(c_1^s, \dots, c_n^s)$. We denote by $\mathrm{val}(x,s) = \sum_{i=1}^{n} c_i^s x_i$ 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 $\mathscr{P}$, denoted by Min–Max $\mathscr{P}$, consists of finding a solution x having the best worst case value across all scenarios, which can be stated as

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

Given a solution $x \in X$, its *regret*, $R(x,s)$, under scenario $s \in S$ is defined as $R(x,s) = \mathrm{val}(x,s) - \mathrm{val}_s^*$. 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 $\mathscr{P}$, denoted by Min–Max Regret $\mathscr{P}$, consists of finding a solution x minimizing the maximum regret $R_{\max}(x)$ which can be stated as

Table 1
Approximation results for min–max and min–max regret versions

	Constant		Non-constant	
	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 $\left(\frac{3}{2}-\varepsilon\right)$ approx.	Not $\left(\frac{3}{2}-\varepsilon\right)$ approx.
Knapsack	fptas	Not at all approx.	Not at all approx.	Not at all approx.

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

When $\mathscr{P}$ is a maximization problem, the max–min and min–max regret versions associated to $\mathscr{P}$ 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 $\mathrm{opt}(I)$ the optimum value of instance I. The *performance ratio* of x is $r(x) = \max\left\{\frac{\mathrm{val}(x)}{\mathrm{opt}(I)}, \frac{\mathrm{opt}(I)}{\mathrm{val}(x)}\right\}$, 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 $r(x) \leqslant f(|I|)$. 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., [1,5,14]). Let $\mathscr{P}$ be a decision problem and $\mathscr{Q}$ a minimization problem. $\mathscr{P}$ is gap-introducing reducible to $\mathscr{Q}$ if there exist a polynomial-time computable function f and a constant $\alpha > 0$ such that, given an instance I of $\mathscr{P}$, it is possible to construct in polynomial time an instance I' of $\mathscr{Q}$, such that

- if I is a positive instance then $\mathrm{opt}(I') \leqslant f(I')$,
- if I is a negative instance then $\mathrm{opt}(I') > \alpha f(I')$.

If $\mathscr{P}$ is an NP-hard problem, and $\mathscr{P}$ is gap-introducing reducible to $\mathscr{Q}$, then $\mathscr{Q}$ is not α-approximable if $P \neq \mathrm{NP}$.

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 $\mathscr{P} \in \mathscr{C}$, denoted by Multi-objective $\mathscr{P}$, has input k objective functions (or criteria) where the hth objective function has coefficients $c_1^h, \ldots, c_n^h$. We denote by $\mathrm{val}(x,h) = \sum_{i=1}^n c_i^h x_i$ 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 $\mathrm{val}(y,h) \leqslant \mathrm{val}(x,h)$ 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 $\mathscr{P}$ 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 $\mathrm{val}(y,h) \leqslant f(n)\mathrm{val}(x,h)$ 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. Constant number of scenarios

3.1. Min–max problems

Consider a minimization problem $\mathscr{P}$. It is easy to see that at least one optimal solution for Min–Max $\mathscr{P}$ is necessarily an efficient solution. Indeed, if $x \in X$ dominates $y \in X$ then $\max_{s\in S} \mathrm{val}(x,s) \leqslant \max_{s\in S} \mathrm{val}(y,s)$. Therefore, we obtain an optimal solution for Min–Max $\mathscr{P}$ by taking, among the efficient solutions, one that has a minimum $\max_{s\in S} \mathrm{val}(x,s)$. Observe, however, that if Min–Max $\mathscr{P}$ 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 $\mathscr{P}$ *has a polynomial-time $f(n)$-approximation algorithm, then* Min–Max $\mathscr{P}$ *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 $\mathscr{P}$ is efficient, there exists a solution $y \in F$ such that $\mathrm{val}(y,s) \leqslant f(n)\mathrm{val}(x^*,s)$, for $s \in S$. Consider among the set F a solution z that has a minimum $\max_{s\in S}\mathrm{val}(z,s)$. Thus, $\max_{s\in S}\mathrm{val}(z,s) \leqslant \max_{s\in S}\mathrm{val}(y,s) \leqslant \max_{s\in S} f(n)\mathrm{val}(x^*,s) = f(n)\mathrm{opt}(I)$. $\square$

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

Proof. Multi-objective versions of shortest path, minimum spanning tree, and knapsack problems have an fptas, for a constant number of criteria, as shown in [11,3]. □

3.2. Min–max regret problems

3.2.1. General results

As for min–max, at least one optimal solution for Min–Max Regret $\mathscr{P}$ is necessarily an efficient solution for Multi-objective $\mathscr{P}$. Indeed, if $x \in X$ dominates $y \in X$ then $\mathrm{val}(x,s) \leqslant \mathrm{val}(y,s)$, for each $s \in S$, and thus $R_{\max}(x) \leqslant R_{\max}(y)$. Therefore, we obtain an optimal solution for Min–Max Regret $\mathscr{P}$ 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.

We establish in the following a sufficient condition for recognizing some particular instances of min–max regret optimization problems.

Theorem 2. *If an optimization problem $\mathscr{P}$ is polynomial-time solvable, then instances I of* Min–Max Regret $\mathscr{P}$ *with* $\mathrm{opt}(I) = 0$ *are recognizable and solvable in polynomial time.*

Proof. Suppose that $\mathscr{P}$ is a minimization problem and consider an instance I of Min–Max Regret $\mathscr{P}$ defined on k scenarios, with a set of scenarios $S = \{s_1, \ldots, s_k\}$. Instance I has $\mathrm{opt}(I) = 0$ if and only if there exists a feasible solution x such that $\mathrm{val}(x,s) = \mathrm{val}^*_s$, for all $s \in S$. This condition is equivalent with the existence of a feasible solution x for instance I' (which is the instance of Multi-objective $\mathscr{P}$ corresponding to I), with value exactly $(\mathrm{val}^*_{s_1}, \ldots, \mathrm{val}^*_{s_k})$. In order to verify this condition, it is sufficient to construct an efficient solution $\bar{x}$ for I'. If $\bar{x}$ has value $(\mathrm{val}^*_{s_1}, \ldots, \mathrm{val}^*_{s_k})$ then $\mathrm{opt}(I) = 0$, otherwise since $\bar{x}$ is efficient, a solution with value $(\mathrm{val}^*_{s_1}, \ldots, \mathrm{val}^*_{s_k})$ does not exist and $\mathrm{opt}(I) > 0$. In order to construct an efficient solution for instance I' of Multi-objective $\mathscr{P}$, we consider any instance I'' of $\mathscr{P}$ with $c_i = \sum_{s \in S} \lambda^s c^s_i$, where λ^s are arbitrary positive weights. An optimum solution of I'' is necessarily an efficient solution for I'. □

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

Theorem 3. *For any polynomial-time solvable minimization problem $\mathscr{P}$ whose feasible solutions have a fixed size and for any function $f : \mathbb{N} \to (1, \infty)$, if* Min–Max $\mathscr{P}'$ *has a polynomial-time $f(n)$-approximation algorithm, then* Min–Max Regret $\mathscr{P}$ *has a polynomial-time $f(n)$-approximation algorithm.*

Proof. Let t be the size of all feasible solutions of any instance of $\mathscr{P}$. Consider an instance I of Min–Max Regret $\mathscr{P}$ where c^s_i 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 $\bar{I}$ of Min–Max $\mathscr{P}'$ with the same number of scenarios, where $\overline{c^s_i} = c^s_i - \frac{\mathrm{val}^*_s}{t}$. Remark that some coefficients could be negative but any feasible solution has a non-negative value. Let $\overline{\mathrm{val}}(x,s)$ denote the value of solution x in scenario s in $\bar{I}$. 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 $R(x,s) = \mathrm{val}(x,s) - \mathrm{val}^*_s = \overline{\mathrm{val}}(x,s)$ since any feasible solution is of size t. Therefore, an optimum solution for I is also an optimum solution for $\bar{I}$ with $\mathrm{opt}(\bar{I}) = \mathrm{opt}(I)$. □

3.2.2. Min–max regret spanning tree

In this section, we prove that Min–Max Regret Spanning Tree has an fptas. For this purpose, we first establish the following preliminary result.

Theorem 4. Multi-objective (Spanning Tree)$'$ *is in FPTAS.*

Proof. Papadimitriou and Yannakakis [11] (Theorems 2 and 4) established a general scheme for constructing fptas for multi-objective versions of several combinatorial optimization problems, including Spanning Tree. However, these results cannot be extended directly to (Spanning Tree)$'$ since non-negative coefficients are required. We can follow similar ideas and obtain an fptas for our problem by making use of a pseudo-polynomial algorithm, based on the matrix-tree theorem that computes a multivariate polynomial of the form

$$\sum_{v_1,\ldots,v_k \in V^{\mathscr{T}}} a_{v_1,\ldots,v_k}, \prod_{h=1}^{k} y_h^{v_h}, \tag{1}$$

where $a_{v_1,\dots,v_k}$ is the number of spanning trees with value v_h on criterion h of a given graph and $V^{\mathscr{T}}$ is the set of values reached on all scenarios, for all spanning trees of G (see [2,6]).

Consider an instance I of Multi-objective (Spanning Tree)$'$ described by a connected graph $G=(V,E)$ with $|V|=n$ and $|E|=m$. Denote by c_{ij}^h the cost of edge (i,j) on criterion h. By Theorem 2 in Papadimitriou and Yannakakis [11], an fptas exists for this problem if and only if there is an fptas for the following gap problem: given an instance and a k-tuple of bounds $(b_1,\dots,b_k)$, either decide if there is a spanning tree T with $\mathrm{val}(T,h)\leqslant b_h$, $h=1,\dots,k$ and construct such a spanning tree if the answer is yes, or answer that there is no spanning tree T' with $\mathrm{val}(T',h)\leqslant \frac{b_h}{1+\varepsilon}$, $h=1,\dots,k$.

In the following we try to solve the gap problem. Let $r=\lceil (n-1)(1+\frac{1}{\varepsilon})\rceil$. Define a new instance where for each edge $(i,j)\in E$, $\bar{c}_{ij}^h=\lceil \frac{c_{ij}^h r}{b_h}\rceil$ if $b_h\neq 0$, and $\bar{c}_{ij}^h=c_{ij}^h$ if $b_h=0$, $h=1,\dots,k$. Let $\overline{\mathrm{val}}(T,h)$ denote the value of a spanning tree T on criterion h using costs $\bar{c}_{ij}^h$. We can easily prove that, for any spanning tree T, if $\overline{\mathrm{val}}(T,h)\leqslant r$ then $\mathrm{val}(T,h)\leqslant b_h$, and if $\overline{\mathrm{val}}(T,h)>r$ then $\mathrm{val}(T,h)>\frac{b_h}{1+\varepsilon}$, for $h\in\{1,\dots,k\}$ such that $b_h\neq 0$. Thus it suffices to determine if there is a spanning tree T with $\overline{\mathrm{val}}(T,h)\leqslant r$, for $h\in\{1,\dots,k\}$ such that $b_h\neq 0$ and $\overline{\mathrm{val}}(T,h)=0$, for $h\in\{1,\dots,k\}$ such that $b_h=0$.

In order to solve this decision problem in polynomial time in $|I|$ and $\frac{1}{\varepsilon}$, it is important to use an algorithm which can compute the multivariate polynomial (1) up to a degree r in each variable y_h, $h=1,\dots,k$. Such an algorithm has been proposed by Mahajan and Vinay [9], with a running time $\mathrm{O}(n^4 r^k\log r)=\mathrm{O}\left(\frac{n^{k+4}}{\varepsilon^k}\log\frac{n}{\varepsilon}\right)$.

When we have a positive answer for this decision problem, we construct such a solution using *self-reducibility* (see, e.g., [10]). It consists of testing iteratively, for each edge if the graph obtained by contracting this edge admits a spanning tree with the required conditions. This requires $\mathrm{O}(m)$ calls to the decision problem.

As shown by Papadimitriou and Yannakakis [11], an $(1+\varepsilon)$-approximation of the set of efficient solutions can be obtained by partitioning the criterion space into $\mathrm{O}\left(\left(\frac{\log nc_{\max}}{\varepsilon}\right)^k\right)$ hyperrectangles, where $c_{\max}=\max_{(i,j)\in E,h=1,\dots,k}c_{ij}^h$. By applying the gap problem to each corner $(b_1,\dots,b_k)$ of all hyperrectangles, which can be solved in time $\mathrm{O}\left(m\frac{n^{k+4}}{\varepsilon^k}\log\frac{n}{\varepsilon}\right)$, and keeping a non-dominated subset of all solutions returned, we obtain an $(1+\varepsilon)$-approximation of the set of efficient solutions in time $\mathrm{O}\left(m\frac{n^{k+4}}{\varepsilon^{2k}}(\log nc_{\max})^k\log\frac{n}{\varepsilon}\right)$. $\square$

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

Proof. Observing that Theorem 1 is also valid for any problem $\mathscr{P}'$, and using Theorems 3 and 4, the result follows.

Notice that a more efficient fptas can be obtained by recording only the values in all criteria of an $(1+\varepsilon)$-approximation of the efficient solutions. The self-reducibility, which is time consuming, is applied only once, for one of the solutions having the minimum maximum regret. The running time of the fptas is thus $\mathrm{O}\left(\frac{n^{k+4}}{\varepsilon^{2k}}(\log nc_{\max})^k\log\frac{n}{\varepsilon}\right)$. $\square$

3.2.3. 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 5. Min–Max Regret Shortest Path, *with a constant 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 $V=\{1,\dots,n\}$ 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 $(\mathrm{val}_s^*)^i$, $s\in S$, $1\leqslant i\leqslant n$ be the value of a shortest path in graph G from 1 to i under scenario s and let $(\mathrm{val}^*)^i$ be the vector of size k of these values $(\mathrm{val}_s^*)^i$, $s\in S$.

In the following, we describe firstly a dynamic programming algorithm that computes at each stage i, $1\leqslant i\leqslant n$, 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 $r_s^i=\mathrm{val}(P_i,s)-(\mathrm{val}_s^*)^i$, $s\in S$. Denote by P_j the path constructed from P_i by adding arc (i,j). The regret of P_j is $r_s^j=\mathrm{val}(P_i,s)+c_{ij}^s-(\mathrm{val}_s^*)^j=r_s^i+(\mathrm{val}_s^*)^i+c_{ij}^s-(\mathrm{val}_s^*)^j$, $s\in S$. The algorithm starts by initializing $R^1=\{(0,\dots,0)\}$, where $(0,\dots,0)$ is a vector of size k and for $2\leqslant j\leqslant n$ let

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

where the operator "Min" preserves the efficient vectors.

Observe that, for $2 \leqslant j \leqslant n$, 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 $2 \leqslant j \leqslant n$ as follows:

$$U^j = \cup_{i\in\Gamma^{-1}(j)}\{r^i + (\mathrm{val}^*)^i + c_{ij} - (\mathrm{val}^*)^j : r^i \in T^i\},$$
$$T^j = Red(U^j),$$

where *Red* is an operator satisfying the following property

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

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

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

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

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 $r = r' + (\mathrm{val}^*)^i + c_{ij} - (\mathrm{val}^*)^j$. Since $(\mathrm{val}^*)^i + c_{ij} \geqslant (\mathrm{val}^*)^j$, we have $r \geqslant r'$. Using the induction hypothesis for i, there exists $\tilde{r} \in T^i$ such that $\tilde{r} \leqslant r'(1+\varepsilon)^{\frac{i-1}{n-1}}$. Since $\tilde{r} \in T^i$ and $(i,j) \in A$, we have $\tilde{r} + (\mathrm{val}^*)^i + c_{ij} - (\mathrm{val}^*)^j \in U^j$ and, using the property satisfied by *Red*, there exists $\bar{r} \in T^j$ such that:

$$\begin{aligned}\bar{r} &\leqslant \left[\tilde{r} + (\mathrm{val}^*)^i + c_{ij} - (\mathrm{val}^*)^j\right](1+\varepsilon)^{\frac{1}{n-1}}\\ &\leqslant [r'(1+\varepsilon)^{\frac{i-1}{n-1}} + r - r'](1+\varepsilon)^{\frac{1}{n-1}} \leqslant r(1+\varepsilon)^{\frac{i}{n-1}}\\ &\leqslant r(1+\varepsilon)^{\frac{j-1}{n-1}}.\end{aligned}$$

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

We show in the following how this algorithm can be implemented in polynomial time in $|I|$ and $\frac{1}{\varepsilon}$. Let $c_{\max} = \max_{(i,j)\in A, s\in S} c^s_{ij}$. For any $s \in S$ and $2 \leqslant j \leqslant n$, we have $r^j_s \leqslant (n-1)c_{\max}$. An operator *Red* can be implemented in polynomial time using the technique of interval partitioning described by Sahni [13]. 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 hyperrectangle of the cartesian product of subintervals. Thus $|T^j| \leqslant \left(\frac{n\log nc_{\max}}{\log(1+\varepsilon)}\right)^k$, $2 \leqslant j \leqslant n$ and the time complexity of our algorithm is $O\left(n\left(\frac{n\log nc_{\max}}{\log(1+\varepsilon)}\right)^k\right)$ 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^ℓ_j which represent the set of efficient vectors of regrets for paths from 1 to j of length at most ℓ, $j = 2,\ldots,n$. □

3.2.4. 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 Partition which is known to be NP-hard [7].

Partition

Input: A finite set A and an integer size $s(a)$ for each $a \in A$.

Question: Is there a feasible partition, i.e. a partition $(A', A\backslash A')$, $A' \subseteq A$ such that $\sum_{a\in A'} s(a) = \sum_{a\in A\backslash A'} s(a)$.

Theorem 6. *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 gap-introducing reduction from Partition. Consider an instance I of Partition characterized by a set $A = \{a_0, a_1, \ldots, a_{n-1}\}$, and a size $s(a)$ for each $a \in A$. 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: $v^1_0 = n^3 d$, $v^1_i = 0$, for $i = 1,\ldots,n$ and

$v_i^2 = n^2 s(a_i)$, for $i = 0, \ldots, n-1$, and $v_n^2 = n^2 d$. Clearly, the optimum value for I' is $n^3 d$ in the first scenario and $n^2 d$ in the second scenario.

Consider that there exists a feasible partition $(A', A \setminus A')$ in I. Suppose that $a_0 \in A'$, otherwise we exchange A' with $A \setminus A'$. In this case, the solution x^* corresponding to A' in I' has $R_{\max}(x^*) = 0$, and thus $\text{opt}(I') = 0$.

Consider now the case where there is no feasible partition in I. If a solution x of I' does not contain a_0 then $R_{\max}(x) = n^3 d$. If a solution x of I' contains a_0 then $R_{\max}(x) \geqslant n^2$ and thus $\text{opt}(I') \geqslant n^2$. □

Remark that in the previous reduction we obtain instances of Min–Max Regret Knapsack with an optimum value equal to 0. Thus, we were able to establish a very strong non-approximability result for this problem. In contrast, all the other min–max regret versions we studied deal with polynomial problems and admit an fptas. It is interesting to observe for the latter problems that, unlike for the knapsack, we can recognize in polynomial-time instances with an optimum value equal to 0 as shown in Theorem 2.

We conclude this section giving some precisions about the complexity status of these problems. Pseudo-polynomial-time algorithms were given by Kouvelis and Yu [8], in the case of a constant number of scenarios, for min–max (max–min) and min–max regret versions of shortest path, knapsack, and minimum spanning tree on grid graphs. Our fptas for min–max and min–max regret spanning tree establish the existence of pseudo-polynomial-time algorithms for these problems on general graphs. Thus min–max (max–min) and min–max regret versions of shortest path, minimum spanning tree and knapsack are weakly NP-hard.

4. Non-constant number of scenarios

When the number of scenarios is non-constant, Kouvelis and Yu [8] proved that min–max and min–max regret shortest path as well as min–max spanning tree and max–min knapsack are strongly NP-hard. We establish the strong NP-hardness of min–max regret knapsack and min–max regret spanning tree in Theorems 7 and 11 respectively.

Concerning approximability results, reductions used by Kouvelis and Yu [8] 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 8–11. On the other hand, the reduction used by Kouvelis and Yu [8] 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 6.

Now we state and prove the above-mentioned results.

Theorem 7. Min–Max Regret Knapsack, *with a non-constant 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 $\text{val}^*_{s_j} = 2$, 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 $\text{val}(x', s_j) \geqslant 1$, for any $s_j \in S$ since edge j is covered by V'. Thus, $R_{\max}(x') \leqslant 1$, which implies $\text{opt}(I) \leqslant 1$.

If G has no vertex cover of size at most k then for any $V' \subseteq V$, $|V'| \leqslant k$, 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 $\text{val}(x', s_j) = 0$, and thus $R_{\max}(x') = 2$, which implies $\text{opt}(I) = 2$.

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 6.

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 (see [4]).

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 8. Min–Max Shortest Path, *with a non-constant 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 $\text{val}(p, s) \leqslant 1$. Then $\max_{s \in S} \text{val}(p, s) \leqslant 1$, which implies $\text{opt}(I') \leqslant 1$.

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 $\text{val}(p, s_h) = 2$ in both cases. Thus $\max_{s \in S} \text{val}(p, s) = 2$, which implies $\text{opt}(I') = 2$. □

Theorem 9. Min–Max Regret Shortest Path, *with a non-constant 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 $V' = V \cup \{n+1, \ldots, n+|S|\}$, $A' = A \cup \{(1, i)\colon\ i = n+1, \ldots, n+|S|\} \cup \{(i, n)\colon\ i = n+1, \ldots, n+|S|\}$, 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 $s \in S$

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

Obviously, $\text{val}_{s_i}^* = 0$ 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 $\text{opt}(I'') \leqslant 1$, otherwise $\text{opt}(I'') = 2$. □

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 $\left(\frac{3}{2} - \varepsilon\right)$-approximable, for any $\varepsilon > 0$ (see [4]).

Minimum Degree Spanning Tree

Input: A graph $G = (V, E)$.

Output: A spanning tree such that its maximum degree is minimum.

Theorem 10. Min–Max Spanning Tree, *with a non-constant number of scenarios, is not* $\left(\frac{3}{2} - \varepsilon\right)$*-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 $\text{val}(T, s_i)$. 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 $\max_{s \in S} \text{val}(T, s)$. □

Theorem 11. Min–Max Regret Spanning Tree, *with a non-constant number of scenarios, is strongly NP-hard. Moreover, it is not* $\left(\frac{3}{2} - \varepsilon\right)$*-approximable, for any* $\varepsilon > 0$*, unless* $P = NP$.

Proof. We construct a gap-introducing reduction from 3SAT (see Fig. 1). 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 three variables, we construct an

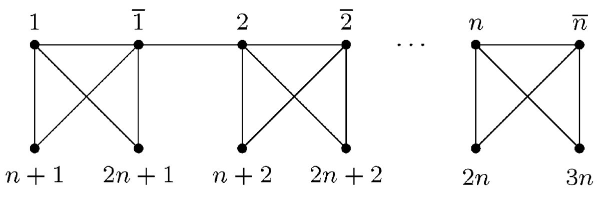

Fig. 1. Min–Max Regret Spanning Tree instance resulting from 3SAT instance.

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, $\overline{i}$, correspond to variable u_i, $i=1,\ldots,n$. Edge set is $E=\{(i,n+i),(i,2n+i),(\overline{i},n+i),(\overline{i},2n+i),(i,\overline{i}):i=,\ldots,n\}\cup\{(\overline{i},i+1):i=1,\ldots,n-1\}$. 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: $c^{s_j}_{i,2n+i}=1$ if $u_i\in C_j$, $c^{s_j}_{\overline{i},2n+i}=1$ 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: $c^{s'_{n+i}}_{i,n+i}=c^{s'_{n+i}}_{\overline{i},n+i}=n$, $c^{s'_{2n+i}}_{i,2n+i}=c^{s'_{2n+i}}_{\overline{i},2n+i}=n$, for every $i=1,\ldots,n$ and 0 otherwise. The values of edges in scenario $s'_j\in S_3$ are defined as follows: $c^{s'_i}_{i,n+i}=c^{s'_i}_{i,2n+i}=c^{s'_i}_{i\overline{i}}=2$, $c^{s'_{\overline{i}}}_{\overline{i},n+i}=c^{s'_{\overline{i}}}_{\overline{i},2n+i}=c^{s'_{\overline{i}}}_{i\overline{i}}=2$, 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 $\{(\overline{i},i+1):i=1,\ldots,n-1\}$ and $\{(i,n+i),(\overline{i},2n+i),(i,\overline{i})\}$, for every i such that $u_i\in C_j$, or $\{(i,2n+i),(\overline{i},n+i),(i,\overline{i})\}$, otherwise. Obviously, this tree has value 0 in scenario s_j. For any scenario $s'_{n+i}\in S_2$, $\text{val}^*_{s'_{n+i}}=n$ since any spanning tree contains one of the edges $(i,n+i),(\overline{i},n+i)$. Similarly, $\text{val}^*_{s'_{2n+i}}=n$, for all $s'_{2n+i}\in S_2$. For any scenario $s'_i\in S_3$, $\text{val}^*_{s'_i}=2$ since any spanning tree contains at least one of the edges $(i,n+i)$, $(i,2n+i)$, $(i,\overline{i})$. Similarly, $val^*_{s'_{\overline{i}}}=2$, for all $s'_{\overline{i}}\in S_3$.

A spanning tree in G necessarily contains edges $(\overline{i},i+1)$, $i=1,\ldots,n-1$. We show in the following that every spanning tree T containing edges $(i,\overline{i})$, $(i,n+i)$ and $(\overline{i},2n+i)$ or edges $(i,\overline{i})$, $(i,2n+i)$ and $(\overline{i},n+i)$ for every $i=1,\ldots,n$, has $R_{\max}(T)\leqslant 3$. Moreover, any other spanning tree T' in G has $R_{\max}\ (T')\geqslant 4$. We have $\text{val}(T,s_j)\leqslant 3$, for any $s_j\in S_1$, $\text{val}(T,s'_j)=n$, for any $s'_j\in S_2$, and $\text{val}(T,s'_j)=4$, for any $s'_j\in S_3$. Thus, $R_{\max}(T)\leqslant 3$. If T' contains both edges $(i,n+i)$, $(\overline{i},n+i)$ for some i, then $\text{val}(T',s'_{n+i})=2n$ and thus $R_{\max}(T')=n$. We can also see that if a spanning tree T' contains both edges $(i,2n+i)$, $(\overline{i},2n+i)$ for some i, then $\text{val}(T',s'_{2n+i})=2n$ and thus $R_{\max}(T')=n$. Consider in the following spanning trees T' that contain edges $(i,\overline{i})$, $i=1,\ldots,n$. If T' contains both edges $(i,n+i)$, $(i,2n+i)$ for some i, then $\text{val}(T',s'_i)=6$ and thus $R_{\max}(T')=4$. We can see also that if T' contains both edges $(\overline{i},n+i)$, $(\overline{i},2n+i)$ for some i, then $\text{val}(T',s'_{\overline{i}})=6$ and thus $R_{\max}(T')=4$. Thus an optimum solution in G is a spanning tree T that contains edges $(\overline{i},i+1)$, $i=1,\ldots,n-1$, edges $(i,\overline{i})$, $i=1,\ldots,n$, and, for every $i=1,\ldots,n$, it contains either edges $(i,n+i)$, $(\overline{i},2n+i)$ or edges $(i,2n+i)$, $(\overline{i},n+i)$. 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 $(i,n+i)$, $(\overline{i},2n+i)$ if and only if u_i takes value 1, and it contains edges $(i,2n+i)$, $(\overline{i},n+i)$ 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, $\text{val}(T,s_j)\leqslant 2$, for any $s_j\in S_1$. Tree T has also $\text{val}(T,s)=n$, for any $s\in S_2$ and $\text{val}(T,s)=4$, for any $s\in S_3$, and thus, $R_{\max}(T)=2$, which implies $\text{opt}(I)=2$.

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 $\text{val}(T,s_j)=3$, and thus $R_{\max}(T)=3$, which implies $\text{opt}(I)=3$. □

Acknowledgements

The authors are grateful to the anonymous referees for their helpful comments and valuable suggestions. This work has been partially funded by grant CNRS/CGRI-FNRS number 18227. The second author was partially supported by the ACI Sécurité Informatique grant-TADORNE project 2004.

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and approximation, Combinatorial Optimization Problems and their Approximability Properties, Springer-Verlag, 1999.

[2] F. Barahona, R. Pulleyblank, Exact arborescences, matching and cycles, Discrete Applied Mathematics 16 (1987) 91–99.

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

[4] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, 1979.

[5] D.S. Hochbaum, Approximation Algorithms for NP-hard Problems, PWS Publishing, Boston, MA, USA, 1997.

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

[7] R.M. Karp, Reducibility among combinatorial problems, Complexity of Computer Computations, Plenum Press, 1972, 85–103.

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

[9] M. Mahajan, V. Vinay, A combinatorial algorithm for the determinant, in: Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1997), New Orleans, USA, 1997, pp. 730–738.

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

[11] C.H. Papadimitriou, 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, 2000, pp. 86–92.

[12] H.M. Safer, J.B. Orlin, Fast approximation schemes for multi-criteria combinatorial optimization. Technical Report 3756-95, Sloan School of Management, 1995.

[13] S. Sahni, General techniques for combinatorial approximation, Operations Research 25 (6) (1977) 920–936.

[14] V.V. Vazirani, Approximation Algorithms, Springer-Verlag, 2001.