Two-stage stochastic matching and spanning tree problems: polynomial instances and approximation

Bruno Escoffier, Laurent Gourvès, Jérôme Monnot, Olivier Spanjaard

Two-stage stochastic matching and spanning tree problems: polynomial instances and approximation

Bruno Escoffier [a] Laurent Gourvès [a] Jérôme Monnot [a,*] Olivier Spanjaard [b]

[a] *LAMSADE-CNRS, Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny, F-75775 Paris Cedex 16, France*

[b] *LIP6-CNRS, Université Pierre et Marie Curie (UPMC), 104 Avenue du Président Kennedy, F-75016 Paris, France*

Abstract

This article deals with the two stage stochastic model, which aims at explicitely taking into account uncertainty in optimization problems, that Kong and Schaefer have recently studied for the maximum weight matching problem [10]. They have proved that the problem is **NP**-hard, and they have provided a factor $\frac{1}{2}$ approximation algorithm. We further study this problem and strengthen the hardness results, slightly improve the approximation ratio and exhibit some polynomial cases. We similarly tackle the maximum weight spanning tree problem in the two stage setting. Finally, we make numerical experiments on randomly generated instances to compare the quality of several interesting heuristics.

Key words: Stochastic programming, Approximation algorithms, Matching, Maximum spanning tree, Combinatorial optimization

1 Introduction

Stochastic optimization aims at explicitely taking into account uncertainty in optimization problems. Uncertainty is modeled by using probability distributions on the parameters of the problem. Two approaches have been investigated: the *open loop* one where a complete solution is computed once for all at the beginning (without knowing anything but the probability distributions on the parameters), and the *closed loop* one where a solution is progressively completed along several stages. In the latter approach, the most widely studied model is two-stage

* Corresponding author.
Email addresses: escoffier@lamsade.dauphine.fr (Bruno Escoffier), gourves@lamsade.dauphine.fr (Laurent Gourvès), monnot@lamsade.dauphine.fr (Jérôme Monnot), olivier.spanjaard@lip6.fr (Olivier Spanjaard).

stochastic optimization (or more generally multi-stage stochastic optimization). We consider here a two-stage model where the values of the different parameters are deterministic in the first stage, and there is a discrete set of scenarios for these values in the second stage. A subset of variables –the choice of which is open– is set during the first stage. During the second stage, after the occurring scenario is known, the remaining variables are set so as to optimize the overall value of the solution. Each of the possible scenarios has a probability to occur, and the goal is to optimize the expected value of the solution built.

More precisely, we consider two stage stochastic versions of classical deterministic problems defined on subsets of edges of a given edge-weighted graph (such as maximum matching problem or maximum spanning tree problem). The underlying problem Π can be defined as follows: We are given a graph G with vertex set V and edge set E. Each edge $e \in E$ has a weight $w(e) \in \mathbb{R}$. Let $\pi : 2^E \mapsto \{true, false\}$ be a given graph property checkable in polynomial time (e.g. being a matching of G). The goal of Π is to find $E' \subseteq E$ such that $\pi(E') = true$ and $\sum_{e \in E'} w(e)$ is maximum.

Then, the two-stage stochastic version of Π, denoted by $\tilde{\Pi}$, is defined as follows. The number of possible scenarios at the second stage is r. We consider uncertainties on edge weights (the underlying graph is fixed). More precisely, we are given $r+1$ weight functions $w_i : E \mapsto \mathbb{R}$, $i = 0, \ldots, r$. Each edge $e \in E$ has a deterministic weight $w_0(e)$ at the first stage. Subsequently, one of the r scenarios occurs, say s, with a known probability p_s. The second stage weight of an edge e when s occurs is $w_s(e)$. A feasible solution consists of choosing a (possibly empty) set E_0 of edges at the first stage, and completing this set E_0 into a feasible solution by choosing a set E_s ($1 \leq s \leq r$) at the second stage. Formally, a feasible solution is a collection of edge sets $(E_0, E_1, \ldots, E_r)$ such that

$$\forall s \in \{1, \ldots, r\},\ E_0 \cap E_s = \emptyset \wedge \pi(E_0 \cup E_s) = true. \tag{1}$$

An edge chosen in the first stage (in E_0) has value $w_0(e)$, while if scenario s occurs, and edge chosen at the second stage (in E_s) has a weight $w_s(e)$. Then the goal is to maximize the expected value $val(E_0, \ldots, E_r)$ of a solution $(E_0, E_1, \ldots, E_r)$, where:

$$val(E_0, \ldots, E_r) = \sum_{e \in E_0} w_0(e) + \sum_{s=1}^{r} \sum_{e \in E_s} p_s\, w_s(e).$$

Throughout the article, to avoid confusion, a solution to the deterministic problem Π is called a Π-solution. A solution to the associated two-stage stochastic problem $\tilde{\Pi}$ is called a $\tilde{\Pi}$-solution. A Π-solution is a set of edges whereas a $\tilde{\Pi}$-solution is a collection of $r+1$ edge sets. Note also that for the ease of presentation for a set of edges E', we sometimes write $w(E')$ instead of $\sum_{e \in E'} w(e)$ (and $w_s(E')$ instead of $\sum_{e \in E'} w_s(e)$).

As mentioned above we focus on two-stage stochastic versions of the maximum weight matching and spanning tree problems. It suffices to give an appropriate definition of π (the graph property) to turn Π and $\tilde{\Pi}$ into the problems under study. For the maximum weight matching problem, π is defined as $\pi(E') = true$ iff $E' \subseteq E$ and edges of E' are pairwise non-adjacent. For the maximum weight spanning tree problem, π is defined as $\pi(E') = true$ iff $E' \subseteq E$ and

$G[E'] = (V, E')$ is a spanning tree of G (in particular, $|E'| = |V| - 1$ and $G[E']$ is acyclic).

1.1 Related work

Stochastic programming is a well established domain in operations research and management (see Kall and Wallace [8], Birge and Louveaux [1] for an introduction). Problems in stochastic programming are typically hard from a computational point of view. Therefore approximation comes naturally. Besides heuristic methods, approximation algorithms (i.e. polynomial time algorithms with a performance guarantee) are valuable tools to tackle hard problems. For a maximization problem, we say that an algorithm is a ρ-approximation if it always returns a solution with value at least ρ times the optimum value ($0 < \rho \leq 1$).

The use of approximation algorithms for problems with uncertain data is quite recent. Here we briefly review recent contributions dealing with approximation algorithms for stochastic versions of well known combinatorial optimization problems.

Ravi and Sinha [13] study several paradigmatic combinatorial problems (shortest path, vertex cover, bin packing, facility location, set cover) in the two-stage stochastic framework and provide approximation algorithms and hardness results. Independently Immorlica, Karger, Minkoff and Mirrokni [7] study two-stage stochastic versions of min cost flow, bin packing, vertex cover, shortest path and Steiner tree.

The problems investigated in [13,7] are characterized by a "monotonicity" property: a partial solution built at the first stage cannot invalidate a possible second stage solution. This is in constrast with the two-stage stochastic versions of the maximum weight matching and maximum cost spanning tree studied in this article. For the matching problem, selecting an edge $[i, j]$ at the first stage precludes all other edges incident to i and j at the second stage. For the spanning tree, selecting $[i, j]$ at the first stage precludes all paths having i and j as endpoints at the second stage.

Gupta, Pàl, Ravi and Sinha [5] propose a general method called *boosted sampling*. Given an approximation algorithm $\mathcal{A}$ for a deterministic problem Π, *boosted sampling* turns $\mathcal{A}$ into an approximation algorithm for the two-stage stochastic version of Π. The authors suppose that Π satisfies a sub-additivity property (the union of two feasible solutions is a solution for the union of the two instances) which is similar to the monotonicty requirement in [7]. Morevover the cost of any resource (e.g. an edge) increases by a given multiplicative factor when the actual scenario is known. The authors deal with the following problems: Steiner tree, vertex cover, facility location, Steiner network. They subsequently generalize *boosted sampling* to multiple stages [6].

Kong and Schaefer [10] introduce the stochastic two-stage maximum matching problem defined upon the well known maximum matching problem (the one studied in this article). They prove that the problem is **NP**-hard when the number of scenarios is an input of the problem, provide a simple and quite general 1/2-approximation algorithm and conduct numerical experiments

on randomly generated instances.

More recently Katriel, Mathieu and Upfal [9] study two stochastic minimum cost maximum matching problems in bipartite graphs. In their two variants, the uncertainty is respectively on the second stage edge cost of the edges and on the set of vertices to be matched. Those problems are abstraction of many real world situations but they are not sub-additive. The authors present approximation algorithms and inapproximability results.

Flaxman, Frieze and Krivelevich [2] study a stochastic minimum spanning tree problem. The first stage price of an edge is a given value in $[0, 1]$. The second stage price is an independent variable uniformly distributed in $[0, 1]$. The problem is to select some edges at both stages which form a tree so that the expected cost at the second stage is minimum. The authors propose a parameterized heuristic and show which value of the parameter optimizes the expected cost. They also generalize their result to the oriented case. Actually the stochastic problem studied by Flaxman *et al* has infinitely many second stage scenarios which are given implicity. For the two-stage stochastic maximum spanning tree problem studied in this article, we have a finite number of explicitely given scenarios.

1.2 Contribution and organization of the article

We study the two-stage stochastic maximum weight matching as introduced by Kong and Schaefer [10] and a similarly defined maximum weight spanning tree problem. Our problems are respectively denoted by 2-Stage max Mat and 2-Stage max ST.

We begin with preliminary remarks in Section 2. Afterwards Section 3 is devoted to the computational complexity of 2-stage max Mat and 2-stage max ST. In particular we strengthen Kong and Schaefer's result by showing that 2-stage max Mat is **APX**-complete even in bipartite graphs of maximum degree 4. Some other classes of graphs (maximum degree 3 and planar) are investigated. The 2-stage max ST is also shown **APX**-complete.

In Section 4 we exhibit polynomial cases of 2-Stage max Mat, namely chains and trees when the number of scenarios is bounded.

An approximation algorithm with performance guarantee $r/(2r-1)$ (where r is the number of possible scenarios at the second stage) is given for problem $\tilde{\Pi}$ in Section 5 (the algorithm being polynomial as soon as Π is polynomially solvable). The result applies for 2-stage max ST and 2-stage max Mat since $\tilde{\Pi}$ generalizes them. Moreover this is an improvement of the 1/2-approximation algorithm of Kong and Schaefer [10] (because $r/(2r-1) > 1/2$). Moreover, for the special case of 2-stage max Mat, we also propose another approximation algorithm with performance guarantee $(\Delta+1)/(2\Delta+1)$ for general graphs and $\Delta/(2\Delta-1)$ for bipartite graphs where Δ is the maximum degree of the graph. It is interesting to notice that these latter ratios do not depend on the number of scenarios.

Sections 6 and 7 deal with practical experiments. We propose in Section 6 a heuristic improving

the approximation algorithm analyzed in Section 5. Then, in Section 7, we compare the behavior of several heuristics including the approximation algorithms of Kong and Schaefer [10] on randomly generated instances. Finally open problems and concluding remarks are given in Section 8.

2 Preliminary remarks

Let us introduce three quite straightforward preliminary remarks. In particular, Claim 2 will be used in the practical experiments in order to preprocess the graph.

Claim 1. Assume that the problem Π *is polynomial. Then the problem* $\tilde{\Pi}$ *is polynomial for* $r = 1$.

Indeed, it reduces to Π on the graph where every edge e is valued by $\max\{w_0(e), w_1(e)\}$.

Note also that the following dominance rule holds (whatever the value of r):

Claim 2. If $\sum_{s=1}^{r} p_s w_s(e) \geq w_0(e)$*, then there exists an optimal* $\tilde{\Pi}$*-solution in which edge* e *is not chosen in the first stage.*

Indeed, for every feasible solution in which edge e is chosen in the first stage, there exists another feasible solution, differing only on edge e, the value of which is at least as good. This can be proved as follows. First, if $\tilde{\Pi}$-solution $(E_0, E_1, \ldots, E_r)$ is feasible and $e \in E_0$, then the $\tilde{\Pi}$-solution $(E_0 \setminus \{e\}, E_1 \cup \{e\}, \ldots, E_r \cup \{e\})$ is feasible. Consequently, choosing edge e in the second stage under all scenarios (instead of in the first stage) has no negative impact on the global feasibility of the solution. Furthermore, the value of the objective function is increased by $(\sum_{s=1}^{r} p_s w_s(e)) - w_0(e) \geq 0$, which concludes the proof.

We conclude by the following observation.

Claim 3. All probabilities p_s *can be set to* $1/r$ *if* $w_s(e)$ *is replaced by* $r\, p_s\, w_s(e)$.

3 Computational complexity

3.1 The case of the maximum spanning tree problem

A *spanning tree* T of a connected graph $G = (V, E)$ with n vertices is a set of edges $T \subseteq E$ such that subgraph (V, T) is connected and acyclic. The problem of computing a spanning tree T of a weighted graph (G, w) maximizing $w(T) = \sum_{e \in T} w(e)$ is known to be polynomial (see for instance [4]) and usually called the *maximum spanning tree* problem. To the best of our knowledge, the *two Stage stochastic maximum spanning tree* problem, denoted 2-Stage max ST,

has not been studied so far. On the other hand, the *two Stage stochastic minimum spanning tree* problem has been proved not $O(\log n)$-approximable unless **P**=**NP** in [2]. Although both problems maximum spanning tree and minimum spanning tree are equivalent from a complexity point of view (modify the weight function $w(e)$ by $w'(e) = w_{\max} - w(e)$ for any edge $e \in E$, where $w_{\max} = \max_{e \in E} w(e)$), this property does not seem to hold for 2-Stage min ST and 2-Stage max ST. Thus, we can not conclude from [2] that 2-Stage max ST is **NP**-hard. Here, we prove that this problem is actually **APX**-complete.

Theorem 1 *2-Stage max ST is **APX**-complete.*

PROOF. The proof will be done via an approximation preserving reduction from the maximum independent set problem, MaxIS for short. An instance $G = (V, E)$ of MaxIS consists in a connected graphs. The goal is to find a subset $V' \subseteq V$ of pairwise non-adjacent vertices that minimizes $|V'|$. Its restriction to connected graphs of maximum degree 3, denoted MaxIS_3 has been shown **APX**-complete in [12].

Let $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$ be an instance of MaxIS_3. We build r and $I = (G', w_i, p_i)$ instance of 2-Stage max ST as follows:

- In $G = (V, E)$, we add a new vertex v_0 and n new edges $[v_0, v_i]$ for $i = 1, \ldots, n$. Let G' be the resulting graph.
- There are m scenarios (i.e., $r = m$) and let $p_i = \frac{1}{m}$ for $i = 1, \ldots, r$ be the probabilities associated to these scenarios.
- The different weight functions are given by: for the new edges, we set $w_0([v_0, v_i]) = 1$ and $w_j([v_0, v_i]) = 0$ for $i = 1, \ldots, n$, $j = 1, \ldots, m$. Finally, for the edges of G, $w_i(e_j) = m$ if $i = j$ and $w_i(e_j) = 0$ otherwise, for $i = 0, \ldots, n$, $j = 1, \ldots, m$.

We claim that there exists an independent set of G of size k if and only if there exists $(T_0, \ldots, T_r)$ feasible solution of I for 2-Stage max ST with value $m + k$.

Let V' be an independent set of G. We build the subgraph G_i for any $i = 1, \ldots, m$ in two steps: firstly, we consider the subgraph of G induced by $V \setminus V'$ and secondly, we contract in this subgraph the edge e_i. Now, we build $(T_0, \ldots, T_r)$ as follows: $T_0 = \{[v_0, v_i] : v_i \in V'\}$ and for any $i = 1, \ldots, r$, $T_i = \{e_i\} \cup E_i$ where E_i is a spanning tree on G_i. Clearly, $T_0 \cup T_i$ is a spanning tree of G' since V' is an independent set of G and by construction $T_0 \cap T_i = \emptyset$. Hence, $(T_0, \ldots, T_r)$ is a feasible solution of I for 2-Stage max ST with value $apx(I)$ and we get:

$$apx(I) = \sum_{i=0}^{r} p_i \, w_i(T_i) = |V'| + \frac{1}{m} m^2 = |V'| + m \tag{2}$$

Conversely, let $(T_0, \ldots, T_r)$ be a feasible solution of I for 2-Stage max ST with value $apx(I)$. Let us prove that we can always assume that the following hold for any $k = 1, \ldots, m$:

(i) $e_k \notin T_0$.
(ii) If $e_k = [v_i, v_j]$, then $\{[v_i, v_0], [v_0, v_j]\} \not\subseteq T_0$.
(iii) $e_k \in T_k$.

For (i), assume that $e_k \in T_0$. We build a feasible solution $(T'_0, \ldots, T'_r)$ by setting $T'_0 = T_0 \setminus \{e_k\}$, and $T'_i = T_i \cup \{e_k\}$ for all $i = 1, \ldots, m$. Obviously, $(T'_0, \ldots, T'_r)$ is feasible, and if val denotes the value of $(T'_0, \ldots, T'_r)$, we get $val = apx(I) + 1$.

For (ii), assume that $\{[v_i, v_0], [v_0, v_j]\} \subseteq T_0$ while $e_k = [v_i, v_j] \in E$. In this case, we construct another feasible solution $(T'_0, \ldots, T'_r)$ by setting $T'_0 = T_0 \setminus \{[v_i, v_0]\}$, and $T'_i = T_i \cup \{e_k\}$ for all $i = 1, \ldots, m$. Using Property (i), we deduce that $(T'_0, \ldots, T'_r)$ is a feasible solution and its value is $apx(I)$.

For (iii), assume that $e_k \notin T_k$. $T_0 \cup T_k \cup \{e_k\}$ contains a cycle μ because of Property (i) and $T_0 \cup T_k$ is a spanning tree of G'. Hence, there exists an edge $e_j \in T_k \cap \mu$ because of Property (i) and (ii). Thus, by setting $T'_k = (T_k \setminus \{e_j\}) \cup \{e_k\}$ and $T'_i = T_i$ for all $i \neq k$, we obtain a feasible solution $(T'_0, \ldots, T'_r)$. Finally, if val denotes the value of $(T'_0, \ldots, T'_r)$, we get $val = apx(I) + 1$.

Using Properties (i) and (iii), we get that $T_0 \subseteq \{[v_0, v_i] : i = 1, \ldots, n\}$ and $w_k(T_k) = 1$ for any $k = 1, \ldots, m$. Hence, $w_0(T_0) = apx(I) - m$. We also deduce that $V' = \{v_i \in V : [v_0, v_i] \in T_0\}$ is an independent set of G because of Property (ii) and we get:

$$|V'| = apx(I) - m \tag{3}$$

Using (2) and (3), we deduce that $opt(I) = opt_{MaxIS}(G) + m \leq 7opt_{MaxIS}(G)$ since G is of maximum degree 3. Moreover, we have $opt_{MaxIS}(G) - |V'| = opt(I) - apx(I)$, which concludes the proof. □

Remark 2 *By slightly modifying the proof of Theorem 1, one can show that 2-Stage max ST remains* ***APX****-complete in graphs of maximum degree 4. Actually, we delete edges* $[v_0, v_i]$ *and we replace vertex* v_0 *by a path on* n *vertices,* u_i *for* $i = 1, \ldots, n$*, and we link each vertex* v_i *to* u_i*. Finally, we set* $w_0([u_i, u_{i+1}]) = 1$ *for* $i = 1, \ldots, n-1$*, and* $w_0([u_i, v_i]) = 1$ *for* $i = 1, \ldots, n$*. It is easy to observe that the path on vertices* u_i *is included in* T_0*, and then* $w_0(T_0) = n + |V'|$ *where* V' *is an independent set of* G'*.*

3.2 The case of the maximum matching problem

A matching M of a graph $G = (V, E)$ is a set of edges $M \subseteq E$ such that any two edges of M are not adjacent. The problem of computing a matching of a weighted graph (G, w) maximizing $w(M) = \sum_{e \in M} w(e)$ is known to be polynomial (see for instance [4]) and called the *maximum matching* problem. The two stage stochastic maximum problem, denoted here by 2-Stage max Mat, has been studied from a complexity and approximation point of view in [10]. In this latter article, the authors show that 2-Stage max Mat is **NP**-hard in bipartite graphs and is

Fig. 1. The gadget $H(e_k)$.

1/2-approximable within polynomial time. Here, we improve this first result by establishing the **APX**-completeness of this problem.

Theorem 3 *2-Stage max Mat is **APX**-complete, even in bipartite graphs of maximum degree 4.*

PROOF. The reduction is quite similar to the one given in Theorem 1 and is also done via a L-reduction from MaxIS$_3$.

Let $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$ be an instance of MaxIS$_3$. We build r and $I = (G', w_i, p_i)$ instance of 2-Stage max Mat as follows:

- From $G = (V, E)$, we add n new vertices u_i for $i = 1, \ldots, n$ and n new edges $[u_i, v_i]$ for $i = 1, \ldots, n$. Moreover, we replace each edge $e_k = [v_i, v_j] \in E$ by the gadget $H(e_k)$ depicted in Figure 1, which consists in adding a new vertex v_{e_k} in edge e_k. Let $G' = (V', E')$ be the resulting graph.
- There are m scenarios (i.e., $r = m$) and let $p_i = \frac{1}{m}$ for $i = 1, \ldots, r$ be the probabilities associated to these scenarios.
- The different weight functions are given by: for the new edges $[u_i, v_i]$, , we set $w_0([u_i, v_i]) = 1$ and $w_j([u_i, v_i]) = 0$ for $i = 1, \ldots, n$, $j = 1, \ldots, m$. Finally, for the edges of G' (ie., $[v_i, v_{e_k}]$ and $[v_{e_k}, v_j]$ where $e_k = [v_i, v_j] \in E$), $w_k([v_i, v_{e_k}]) = w_k([v_{e_k}, v_j]) = m$ if $e_k = [v_i, v_j] \in E$ and $w_k(e') = 0$ with $e' \in E' \setminus \{[v_i, v_{e_k}], [v_{e_k}, v_j]\}$ otherwise, for $k = 1, \ldots, m$.

By construction G' is bipartite (using gadget $H(e_k)$) and of maximum degree 4 since G has a maximum degree 3.

We claim that there exists an independent set of G of size k if and only if there exists a feasible solution $(M_0, \ldots, M_r)$ of I for 2-Stage max Mat with value $m + k$.

Let V' be an independent set of G. For M_0 we set $M_0 = \{[u_i, v_i] : v_i \in V'\}$ and for any $k = 1, \ldots, m$ $M_k = \{[v_j, v_{e_k}]\}$ where $e_k = [v_i, v_j]$ and $v_j \notin V'$ (if both v_i, v_j are not in V' we arbitrarily pick one of them). By construction $M_0 \cup M_k$ is a matching of G'. Hence, $(M_0, \ldots, M_r)$ is a feasible solution of I for 2-Stage max Mat with value $ap(I)$ and we get:

$$apx(I) = |V'| + m \tag{4}$$

Conversely, let $(M_0, \ldots, M_r)$ be a feasible solution of I for 2-Stage max Mat with value $apx(I)$. Let us prove that we can always assume that the following holds:

(*i*) $M_0 \subseteq \{[u_i, v_i] : i = 1, \ldots, n\}$.

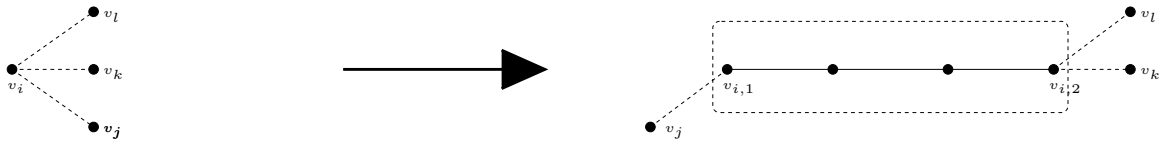

Fig. 2. The gadget $H(v_i)$.

(ii) For any $k = 1, \ldots, m$, if $e_k = [v_i, v_j] \in E$, then $\{[u_i, v_i], [u_j, v_j]\} \not\subseteq M_0$.
(iii) For any $k = 1, \ldots, m$, if $e_k = [v_i, v_j] \in E$, then either $M_k = \{[v_i, v_{e_k}]\}$ or $M_k = \{[v_{e_k}, v_j]\}$.

For (i), by setting $M'_0 = M_0 \cap \{[u_i, v_i] : i = 1, \ldots, n\}$ and $M'_k = M_k$ for any $k = 1, \ldots, m$, we obtain another feasible solution $(M'_0, \ldots, M'_r)$ with the same value as $(M_0, \ldots, M_r)$.

For (ii), assume that $\{[u_i, v_i], [u_j, v_j]\} \subseteq M_0$ and $e_k = [v_i, v_j] \in E$. We build another feasible solution $(M'_0, \ldots, M'_r)$ by setting $M'_0 = M_0 \setminus \{[u_i, v_i]\}$, $M'_k = \{[v_i, v_{e_k}\}$ and $M'_l = M_l$ for all $l = 1, \ldots, m$, $j \neq k$. Obviously, $(M'_0, \ldots, M'_r)$ is a feasible solution and $val(M'_0, \ldots, M'_r) = apx(I)$. By repeating this procedure while it is possible, Property (ii) holds.

For (iii), let $e_k = [v_i, v_j] \in E$. From Property (ii), we know that $[u_i, v_i] \notin M_0$ (or $[u_j, v_j] \notin M_0$). Thus, by setting $M'_l = M_l$ for any $l = 0, \ldots, m$, $l \neq k$, we obtain a feasible solution $(M'_0, \ldots, M'_r)$. Moreover, by construction $val(M'_0, \ldots, M'_r) \geq apx(I)$. In conclusion, by operating this modification while it is possible, Property (iii) holds.

Now, let $(M_0, \ldots, M_r)$ be a feasible solution with value $apx(I)$ and satisfying Properties (i), (ii), and (iii). From Property (iii), we get $apx(I) = w_0(M_0) + m$. Moreover, $V' = \{v_i : [u_i, v_i] \in M_0\}$ is an independent set because of Property (ii). Finally, $|V'| = w_0(M_0) = |M_0|$ from Property (i). In conclusion, we obtain:

$$|V'| = apx(I) - m \tag{5}$$

Since equalities (4) and (5) are the same as in Theorem 1, and since the reduction starts from the same problem, we obtain the expected result. □

Theorem 4 *2-Stage max Mat is **APX**-complete, even in graphs of maximum degree 3.*

PROOF. The proof is a slight modification of the one given in Theorem 3. Instead of adding vertices u_i and edges $[u_i, v_i]$ to G for all $v_i \in V$, we make a local replacement of v_i by a gadget $H(v_i)$. $H(v_i)$ is a path of length 3 with endpoints $v_{i,1}$ and $v_{i,2}$, see Figure 2 for an illustration, where v_j, v_k, v_l are the neighbors of v_i in G. If v_i has a degree less than three, we always delete edge $[v_{i,1}, v_j]$ (thus, we assume that v_k, v_l are the neighbors of v_i in G) and maybe we delete edge $[v_{i,2}, v_k]$ if the degree of v_i in G is one.

The weight of the edges in this path has a cost 1 in scenario 0 and 0 in scenario k for all $k = 1, \ldots, r$.

The rest of the reduction is similar to the one given in Theorem 3. From this way, we obtain a graph G' which has a maximum degree 3. Now, the important point to observe is that we can always assume that the matching M_0 takes for each gadget $H(v_i)$, either 2 edges (the extreme edges of the path of length 3) or one edge (the middle edge of the path). Thus, the equality (5) becomes now:

$$|V'| = apx(I) - m - n \tag{6}$$

Finally, since G has a maximum degree 3, we deduce $n \leq 3m$. Thus, $opt(I) = opt_{MaxIS}(G) + m + n \leq opt_{MaxIS}(G) + 4m \leq 10 opt_{MaxIS}(G)$. The rest of the proof is similar to Theorem 3. □

When the graph is planar, MaxIS$_3$ is known to be **NP**-hard [3]. Since the reductions given in Theorem 3 and Theorem 4 conserve the planarity of the graph, we conclude:

Corollary 1 *2-Stage max Mat is **NP**-hard, even in planar graphs of maximum degree 3, or planar bipartite graphs of maximum degree 4.*

4 Polynomial instances

4.1 Dynamic programming in chains

We now show that 2-stage max Mat can be solved in polynomial time in chains, even with an unbounded number of scenarios. Given a chain denoted by G, we assume that the vertices are numbered from 1 to n when traversing G from left to right. The subgraph of G on vertices $\{i, \ldots, j\}$ is denoted by $G(i, j)$. In the resolution method, the set of feasible solutions is partitioned into two subsets:

(1) subset $\mathcal{S}_1$ of solutions including no edge chosen in the first stage,
(2) subset $\mathcal{S}_2$ of solutions including at least one edge chosen in the first stage.

The handling of subset $\mathcal{S}_1$ is very basic: an optimal solution in $\mathcal{S}_1$ is computed by solving r maximum matching problems (one problem for each scenario). More formally, if we denote by opt_1 the value of an optimal solution in $\mathcal{S}_1$, we have:

$$opt_1 = \sum_{s=1}^{r} p_s z_2^s$$

where z_2^s is the value of an optimal matching on graph G under scenario s.

The handling of subset $\mathcal{S}_2$ is more elaborated, and relies on dynamic programming. For simplicity, we only present here how to compute the *value* of an optimal solution, but the solution

itself can be easily determined by using standard bookkeeping techniques that do not affect the complexity of the algorithm. Let $\mathcal{S}_2^k \subseteq \mathcal{S}_2$ denote the subset of solutions where $[k-1,k]$ is the rightmost edge selected in the first stage. The method is based on the following idea: the value of an optimal solution in $\mathcal{S}_2^k$ equals the value $W(k)$ of an optimal solution on $G(1,k)$ among solutions that includes edge $[k-1,k]$ in the first stage, plus the expected weight in the second stage when choosing a maximal matching on $G(k+1,n)$ under every scenario. Consequently, the value opt_2 of an optimal solution in $\mathcal{S}_2$ equals:

$$opt_2 = \max_{2 \leq k \leq n} \left(W(k) + \sum_{s=1}^{r} p_s z_2^s(k+1,n) \right)$$

where $z_2^s(k+1,n)$ is the value of an optimal matching on subgraph $G(k+1,n)$ under scenario s, and $z_2^s(k+1,n) = 0$ if $k+1 \geq n$. The computation of $W(k)$ is performed in a recursive manner. Two cases must be distinguished: either (i) edge $[k-1,k]$ is the only edge chosen in the first stage, or (ii) there exists at least one other edge in $G(1,k-2)$ that is chosen in the first stage. Let us denote by $W_1(k)$ and $W_2(k)$ the value of an optimal solution in cases (i) and (ii) respectively. In case (i), the equation defining $W_1(k)$ writes:

$$W_1(k) = w_0([k-1,k]) + \sum_{s=1}^{r} p_s z_2^s(1,k-2) \qquad (k \geq 2) \tag{7}$$

In case (ii), if edge $[j-1,j]$ is the rightmost first stage edge on $G(1,k-2)$ in a solution S on $G(1,k)$, then S is compounded of a subsolution on $G(1,j)$ with edge $[j-1,j]$ chosen in the first stage, a subsolution on $G(j+1,k-2)$ with only edges chosen in the second stage, and edge $[k-1,k]$. The recurrence relation defining $W_2(k)$ directly follows:

$$W_2(k) = w_0([k-1,k]) + \max_{2 \leq j \leq k-2} \left(W(j) + \sum_{s=1}^{r} p_s z_2^s(j+1,k-2) \right) \qquad (k \geq 4) \tag{8}$$

Finally, the value of an optimal solution on $G(1,k)$ (under the constraint that edge $[k-1,k]$ is chosen in the first stage) is therefore:

$$W(k) = \max(W_1(k), W_2(k))$$

For initialization, one sets $W_1(2) = w_0([1,2])$, $W_2(2) = -\infty$, $W_1(3) = w_0([2,3])$ and $W_2(3) = -\infty$.

The value of an optimal solution of 2-stage max Mat in a chain is of course $\max\{opt_1, opt_2\}$.

Example 1 *Consider the chain of Figure 3 with two scenarios of equal probabilities. Every edge e is valued by a vector $(w_0(e), w_1(e), w_2(e))$. We have $opt_1 = \frac{1}{2}8 + \frac{1}{2}9 = 8.5$ since $z_2^1 = 8$ and $z_2^2 = 9$. We now need the $W(k)$'s to compute opt_2. The dynamic programming process is indicated in Table 1. For the sake of brevity, we only detail here the computation at node 7. For the computation of $W_1(7)$, one proceeds as follows: the value of an optimal matching on $G(1,5)$ is 6 under scenario 1 ($z_2^1(1,5) = 6$), and it is also 6 under scenario 2 ($z_2^2(1,5) = 6$). The expected value in the second stage is therefore 6 and consequently $W_1(7) = w_0([6,7]) + 6 = 12$. For the computation of $W_2(7)$, one proceeds as follows: the rightmost first stage edge (edge*

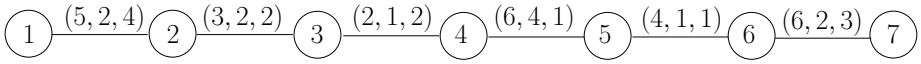

Fig. 3. A two-stage stochastic matching problem on a chain.

k	$W_1(k)$	$W_2(k)$	$W(k)$
2	5	$-\infty$	5
3	3	$-\infty$	3
4	$2+\frac{1}{2}6=5$	$2+\max\{5\}+0=7$	7
5	$6+\frac{1}{2}2+\frac{1}{2}4=9$	$6+\max\{5,3\}=11$	11
6	$4+\frac{1}{2}3+\frac{1}{2}6=8.5$	$4+\max\{3+\frac{1}{2}1+\frac{1}{2}2,3,7\}=11$	11
7	$6+\frac{1}{2}6+\frac{1}{2}6=12$	$6+\max\{5+\frac{1}{2}4+\frac{1}{2}2,3+\frac{1}{2}4+\frac{1}{2}1,7,11\}=17$	17

Table 1
Dynamic programming table for Example 1.

$[6,7]$ *excepted) is either* $[1,2]$, $[2,3]$, $[3,4]$ *or* $[4,5]$. *If the rightmost first stage edge is* $[1,2]$, *then the optimal value is* $6+W(2)+\frac{1}{2}4+\frac{1}{2}2=14$ *since* $z_2^1=4$ *(an optimal matching in* $G(3,5)$ *under scenario 1 is edge* $[4,5]$*), or* $z_2^2=2$ *(an optimal matching in* $G(3,5)$ *under scenario 2 is edge* $[3,4]$*). Similarly, if the rightmost first stage edge is* $[2,3]$, *then the optimal value is* $6+W(3)+\frac{1}{2}4+\frac{1}{2}1=11.5$ *since one completes with edge* $[4,5]$ *under scenario 1 and 2 (optimal matching in* $G(4,5)$*). When edge* $[3,4]$ *or* $[4,5]$ *is the rightmost first stage edge, then graph* $G(5,5)$ *(resp.* $G(6,5)$*) has no edge and thus the optimal value is* $6+W(4)=13$ *(resp.* $6+W(5)=17$*). As displayed in the corresponding cell of Table 1, the value of* $W_2(7)$ *is therefore* $\max\{14,11.5,13,17\}=17$. *The determination of* opt_2 *now requires to compute* $\frac{1}{2}z_2^1(k+1,7)+\frac{1}{2}z_2^2(k+1,7)$ *(*$k=2,\ldots,7$*). One obtains* 5.5 *for* $k=2$, 5 *for* $k=3$, 2.5 *for* $k=4$, 2.5 *for* $k=5$ *and* 0 *for* $k=6,7$. *The overall value is* $opt_2=\max\{5+5.5,3+5,7+2.5,11+2.5,11+0,17+0\}=17$. *The value of an optimal solution in* G *is finally* $\max\{opt_1,opt_2\}=\max\{8.5,17\}=17$. *The corresponding optimal solution is* $(M^0,M^1,M^2)=(\{[1,2],[4,5],[6,7]\},\emptyset,\emptyset)$.

Proposition 2 *Let* $W^*(k)$ *denote the value of an optimal solution in* $G(1,k)$. *We have:* $W^*(k)=W(k)$.

PROOF. Consider subgraph $G(1,k)$. We distinguish between case (ii) and case (i).

(ii) Consider the subset $\mathcal{S}$ of solutions in $G(1,k)$ where edges $[j-1,j]$ and $[k-1,k]$ are chosen in the first stage (and no other edges between them). Clearly, every feasible solution in $\mathcal{S}$ can be partitioned into three parts: the subsolution on $G(1,j)$, the subsolution on $G(j+1,k-2)$ (edges $[j,j+1]$ and $[k-2,k-1]$ cannot be chosen in the first or second stage), and edge $[k-1,k]$. Note that the matchings on $G(1,j)$ and $G(j+1,k-2)$ are independent as soon as edge $[j-1,j]$ is assumed to be chosen in the first stage, i.e. the choice of an edge in $G(1,j)$ has no impact on the feasibility of the choice of an edge in $G(j+1,k-2)$, and conversely. Consequently, the projection on $G(1,j)$ (resp. $G(j+1,k-2)$) of an optimal solution in $\mathcal{S}$ is an optimal solution on $G(1,j)$ (resp. $G(j+1,k-2)$). Thus,

the marginal contribution of the subsolution on $G(1,j)$ to an optimal solution on $G(1,k)$ is $W(j)$. In addition, since the part on $G(j+1,k-2)$ includes only edges chosen in the second stage, the edge chosen under scenario s in an optimal solution corresponds to a maximum-weight matching on graph $G(j+1,k-2)$ valued by w_s, the value of which is $z_2^s(j+1,k-2)$. The marginal contribution of the subsolution on $G(j+1,k-2)$ to an optimal solution on $G(1,k)$ is therefore $\sum_{s=1}^{r} p_s z_2^s(j+1,k-2)$. Finally, the marginal contribution of edge $[k-1,k]$ is of course $w_0([k-1,k])$. Equation (8) follows.

(i) The proof is similar to the one of (ii), except that the solution is partitioned into two parts: the subsolution on $G(1,k-2)$ and edge $[k-1,k]$. □

The polynomiality of the dynamic programming procedure directly follows from the polynomial number (rn^3) of maximal matching routines performed by the algorithm. However, a simpler procedure makes it possible to determine in polynomial time an optimal solution in a tree *when the number of scenarios is bounded*. This is the topic of the next subsection.

4.2 Dynamic programming in trees

Consider a rooted tree T, the root of which is denoted by t. We denote by T_v the subtree rooted at v, and by b a $(r+1)$-tuple of booleans (at v). Tuple b will be dedicated to indicate the status (matched or unmatched) of vertex v in the different scenarios. Thus, b takes value in a subset B of $\{0,1\}^{r+1}$, defined as follows: $B = B_1 \cup B_2$ where $B_1 = \{0\} \times \{0,1\}^r$ (v is unmatched by the first-stage matching) and $B_2 = \{1\} \times \{0\}^r$ (v is matched by the first-stage matching).

The basic idea of the dynamic programming procedure is to compute recursively the value $W(v,b)$ of an optimal solution in $\mathcal{S}(v,b)$, that denotes the subset of feasible solutions in T_v such that v is matched (resp. unmatched) in T_v by the first-stage matching if $b_0 = 1$ (resp. $b_0 = 0$), and v is matched (resp. unmatched) in T_v by the second-stage matching under scenario s if $b_s = 1$ (resp. $b_s = 0$).

The set of values $\{W(v,b) : b \in B\}$ is stored in a table associated to node v. At a leaf ℓ of tree T, the recursion is initialized by setting $W(\ell,b) = 0$ for $b = \mathbf{0}$, and $W(\ell,b) = -\infty$ for $b \in B \setminus \{\mathbf{0}\}$, where $\mathbf{0}$ is the $(r+1)$-tuple the components of which are all equal to zero.

The recursive case can be described as follows. Assume vertex v has a set $\Gamma(v) = \{u_1, \ldots, u_k\}$ ($k > 0$) of children. Given a vector b of booleans, we denote by $S(b)$ the set of s in $\{0,\ldots,r\}$ for which $b_s = 1$. Furthermore, we denote by $\Pi(b,v)$ the set of mappings from $S(b)$ to $\Gamma(v)$. For $\pi \in \Pi(b,v)$ and $s \in S(b)$, edge $[v,\pi(s)]$ is chosen in the first-stage matching if $s = 0$, or in the second-stage matching under scenario s otherwise. Note that $\Pi(b,v) = \emptyset$ when $S(b) = \emptyset$. The recurrence relation is written as follows (where $p_0 = 1$ for convenience):

$$W(v,b) = \max_{\pi \in \Pi(b,v)} \left(\sum_{s \in S(b)} p_s w_s([v,\pi(s)]) + \sum_{j=1}^{k} \max \left\{ W(u_j,b') : b' \in P_{\pi,u_j} \right\} \right) \qquad (9)$$

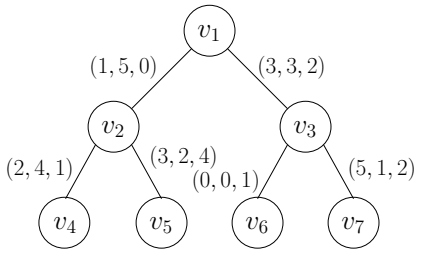

Fig. 4. A two-stage stochastic matching problem on a tree.

where P_{π,u_j} denotes the set of possible tuples b' for u_j compatible with π. More precisely, if edge $[u_j, v]$ is chosen at stage 1 then only $b' = \mathbf{0}$ is possible, while otherwise for any scenario s where $[u_j, v]$ is chosen we have $b'_0 = 0$ and $b'_s = 0$. By convention, $W(v, b) = -\infty$ when $\Pi(b, v) = \emptyset$. The value of the optimal solution is finally obtained at the root by computing $\max_{b \in B} W(t, b)$. Here again, the optimal solution itself can be recovered easily by using standard bookkeeping techniques. For the convenience of the reader, we now give an example of an execution of this dynamic programming algorithm.

Example 3 *Consider the tree of Figure 4. Every edge e is valued by a vector $(w_0(e), w_1(e), w_2(e))$. The probability of scenario 1 is 0.4 and the one of scenario 2 is 0.6. The dynamic programming tables are indicated in Table 2. For the sake of brevity, we only detail here the computation at node v_1, once tables at nodes v_2 and v_3 are known. For $b = (0,0,0)$, we have that $\max_{b' \in B} W(v_2, b') + \max_{b' \in B} W(v_3, b') = W(v_2, (0,1,1)) + W(v_3, (1,0,0)) = 4 + 5 = 9$. For $b = (1,0,0)$, one compares $w_0([v_1, v_2]) + W(v_2, (0,0,0)) + \max_{b' \in B} W(v_3, b) = 1 + 0 + 5 = 6$ (edge $[v_1, v_2]$ is chosen in the first stage) with $w_0([v_1, v_3]) + W(v_3, (0,0,0)) + \max_{b' \in B} W(v_2, b) = 3 + 0 + 4 = 7$ (edge $[v_1, v_3]$ is chosen in the first stage). For $b = (0,1,0)$, one compares $p_1 w_1([v_1, v_2]) + \max\{W(v_2, (0,0,0)), W(v_2, (0,0,1))\} + \max_{b' \in B} W(v_3, b') = 0.4 \times 5 + \max\{0, 2.4\} + 5 = 9.4$ (edge $[v_1, v_2]$ is chosen in the second stage under scenario 1) with $p_1 w_1([v_1, v_3]) + \max\{W(v_3, (0,0,0)), W(v_3, (0,0,1))\} + \max_{b' \in B} W(v_2, b') = 0.4 \times 3 + \max\{0, 1.2\} + 4 = 6.4$ (edge $[v_1, v_3]$ is chosen in the second stage under scenario 1). For $b = (0,0,1)$, by proceeding similarly, one obtains $W(v_1, (0,0,1)) = 6.6$. Finally, for $b = (0,1,1)$, one considers four cases: edge $[v_1, v_2]$ is chosen in the second stage whatever scenario occurs, edge $[v_1, v_3]$ is chosen in the second stage whatever scenario occurs, edge $[v_1, v_2]$ is chosen under scenario 1 while edge $[v_1, v_3]$ is chosen under scenario 2, edge $[v_1, v_3]$ is chosen under scenario 1 while edge $[v_1, v_2]$ is chosen under scenario 2. One obtains $W(v_1, (0,1,1)) = 7$ as a maximum. The value of the optimal solution is therefore $\max\{9, 7, 9.4, 6.6, 7\} = 9.4$. The corresponding optimal solution is $(M^0, M^1, M^2) = (\{[v_3, v_7]\}, \{[v_1, v_2]\}, \{[v_2, v_5]\})$.*

Note that, by applying the dominance rule given in Section 2 (Claim 2), one could have seen that only edges $[v_1, v_3]$ or $[v_3, v_7]$ can be chosen in the first stage in the previous example. We now state formally the result establishing the correctness of the algorithm:

Proposition 4 *Let v be a vertex and $W^*(v, b)$ denote the value of an optimal solution in $\mathcal{S}(v, b)$. We have: $W^*(v, b) = W(v, b)$.*

PROOF. The main ideas of the proof are quite classic. We therefore give here only an outline

b_0	b_1	b_2	$W(v_2, b)$
0	0	0	0
1	0	0	3
0	1	0	1.6
0	0	1	2.4
0	1	1	4

b_0	b_1	b_2	$W(v_3, b)$
0	0	0	0
1	0	0	5
0	1	0	0.4
0	0	1	1.2
0	1	1	1.6

b_0	b_1	b_2	$W(v_1, b)$
0	0	0	9
1	0	0	7
0	1	0	9.4
0	0	1	6.6
0	1	1	7

Table 2
Dynamic programming tables in Example 3.

of the proof. Equation 9 encompasses the two following cases for computing $W(v, b)$:

(i) Assume that $b \in B_2$. It means that an edge $[v, u_h]$ is chosen in the first stage. Then, other edges incident to v cannot be chosen in the first or the second stage. The matching on T_v can therefore be partitioned between $[v, u_h]$ and matchings on trees $T_{u_1}, \ldots, T_{u_k}$. Note that the matchings on $T_{u_1}, \ldots, T_{u_k}$ are independent once $[v, u_h]$ is chosen, i.e. the choice of an edge in T_i has no impact on the feasibility of the choice of an edge in T_j. Hence, by considering all possible edges $[v, u_h]$, we have:

$$W^*(v, (1, 0, \ldots, 0)) = \max_{h \in \{1,\ldots,k\}} \left(w_0([v, u_h]) + W^*(u_h, \mathbf{0}) + \sum_{i \neq h} \max\{W^*(u_i, b') : b' \in B\} \right)$$

(ii) Assume that $b \in B_1$. It means that an edge $[v, u_h]$ is chosen for each scenario s such that $b_s = 1$ (possibly with repeated edges). We denote by H the set of edges chosen in at least one scenario, and by π the mapping from $S(b)$ to H indicating the edge chosen for each scenario. Similarly to the previous case, the matchings on $T_{u_1}, \ldots, T_{u_k}$ are independent once couple (H, π) is set. Hence, by considering all possible (H, π), we have:

$$W^*(v, b) = \max_{H \in \{1,\ldots,k\}} \max_{\pi \in H^{S(b)}} \left(\sum_{h \in H} \sum_{s \in \pi^{-1}(h)} p_s w_s([v, u_h]) + \sum_{h \in H} \max\{W^*(u_h, b) : b' \in P_{\pi,h}\} + \sum_{i \in \Gamma(v) \setminus H} \max\{W^*(u_i, b) : b' \in B\} \right)$$

where $\pi^{-1}(h)$ denotes the set of scenarios in which edge $[v, u_h]$ is chosen, and $P_{\pi,h} = \{b' \in B : b'_0 = 0 \text{ and } b'_s = 0 \; \forall s \in \pi^{-1}(h)\}$. □

This dynamic programming procedure makes it possible to solve the two-stage stochastic matching problem on trees in polynomial time:

Proposition 5 *The two-stage stochastic matching problem can be solved on trees in time $O(rn4^r\Delta^{r+2})$, where Δ is the maximum degree of a vertex.*

PROOF. The soundness of the procedure follows from what has been said before. Dealing with the computation time, there are n tables to compute, each one with $O(2^r)$ lines. In order to compute the rightmost cell of a line (i.e, $W(v,b)$), we have to compare $|\Pi(b,v)| \in O(\Delta^{r+1})$ numbers (the number of mappings from $S(b)$ to $\Gamma(v)$). Each number is computed by summing at most Δ values (summation for j from 1 to k). Each value requires itself to compare $O(2^r)$ elements $W(u_j, b')$, each element being computed in $O(r)$ (to check if b' is in P_{π,u_j}). Hence, the overall complexity follows. □

5 Approximation algorithms

The 1/2-approximation algorithm of Kong and Schaefer [10] for 2-stage max Mat is based on the following idea: one considers 1) a *first-stage myopic* $\tilde{\Pi}$*-solution* where a complete Π-solution is selected during the first stage and no edge is added during the second stage, and 2) a *second-stage myopic* $\tilde{\Pi}$*-solution* where no edge is selected during the first stage and a complete Π-solution is selected during the second stage (according to which scenario occurs). The optimal first-stage myopic $\tilde{\Pi}$-solution can be computed in polynomial time by applying an optimal algorithm for problem Π on the graph valued by w_0, while the optimal second-stage myopic $\tilde{\Pi}$-solution can also be computed in polynomial time by applying an optimal algorithm for problem Π on the graph valued by w_i for $i = 1, \ldots, r$. On the overall, $(r+1)$ executions of an optimal algorithm for problem Π are therefore required. By returning the solution (among two) with the largest value of the objective function, Kong and Schaefer show that one achieves an approximation ratio $\frac{1}{2}$. We now show that the following modification of 1) makes it possible to improve the approximation ratio: instead of a first-stage myopic $\tilde{\Pi}$-solution, one considers a *globally-feasible* $\tilde{\Pi}$*-solution*, such that the union of all the edges selected in the first or second stage (under the different scenarios) is a feasible Π-solution. In other words, we improve the solution computed in step 1) and obtain this way a slightly better approximation ratio. The improvement is interesting when the number of scenarios is small (see Theorem 6), as well as in some particular cases such as for 2-Stage max Mat in bounded degree graphs (Theorem 7).

Our algorithm works for a large class of two-stage stochastic problems. Therefore we present it on a general stochastic problem $\tilde{\Pi}$ defined over the deterministic Π problem (see the introduction for the definition of Π and $\tilde{\Pi}$). We suppose that we have in hand a polynomial time algorithm $\mathcal{A}_\Pi$ associated with Π. The algorithm optimally completes a given partial solution for problem Π. Formally, it takes as inputs an edge-weighted graph $G = (V, E, w)$, instance of Π, and a set $E'' \subseteq E$. If the set $\mathcal{F} = \{E' : E \supseteq E' \supseteq E'' \wedge \pi(E') = true\}$ is nonempty then $\mathcal{A}_\Pi$ returns an element of $\mathcal{F}$, say E^*, which maximizes $\sum_{e \in E^*} w(e)$. Otherwise $\mathcal{A}_\Pi$ states that $\mathcal{F} = \emptyset$.

It is not difficult to adapt the known polynomial algorithms for maximum weight matching and maximum weight spanning tree to obtain an algorithm $\mathcal{A}_\pi$ having the properties mentioned above.

Let us consider the following algorithm `2StageApx` for $\tilde{\Pi}$:

(1) Use $\mathcal{A}_\Pi$ to find a maximum weight Π-solution E^1 in the graph with weights $w^1(e) = \max(w_0(e), \sum_{i=1}^r p_i w_i(e))$. This Π-solution can be turned into a $\tilde{\Pi}$-solution $S^1 = (E_0^1, \cdots, E_r^1)$ by considering that an edge of E^1 is put in E_0^1 if its weight is $w_0(e)$, and put in all the E_i's, $i = 1, \ldots, r$ otherwise.
(2) For any $i = 1, \ldots, r$, find a maximum weight Π-solution E_i^2 in the graph with weights w_i. These solutions form another $\tilde{\Pi}$-solution $S^2 = (\emptyset, E_1^2, \cdots, E_r^2)$.
(3) Return the best $\tilde{\Pi}$-solution S between S^1 and S^2.

Theorem 6 *Algorithm* `2StageApx` *achieves an approximation ratio* $\frac{r}{2r-1}$ *for the two-stage stochastic maximum-weight* Π *problem.*

PROOF. First, notice that by definition, the $\tilde{\Pi}$-solution S^1 computed in the first step of Algorithm `2StageApx` is such that $val(S^1) = w^1(E^1)$ (more precisely $val(S^1) \geq w^1(E^1)$ if edges are added to complete the partial solution).

Now, let $S^* = (E_0^*, E_1^*, \ldots, E_r^*)$ be an optimal $\tilde{\Pi}$-solution, the value of which is denoted by $val(S^*)$.

Remark that $E_0^* \cup E_i^*$ is a Π-solution. Since E^1 is a maximum weight Π-solution in G weighted by w^1, we get:

$$val(S^1) = w^1(E^1) \geq w^1(E_0^*) + w^1(E_i^*) \tag{10}$$

Making the sum of Equation (10) for $i = 1, \ldots, r$, and using the fact that $w^1(e) \geq \max(w_0(e), p_i w_i(e))$, we obtain:

$$\begin{aligned} r \times val(S^1) &\geq r \times w^1(E_0^*) + \sum_{i=1}^r w^1(E_i^*) \\ &\geq r \times w_0(E_0^*) + \sum_{i=1}^r p_i w_i(E_i^*) \end{aligned} \tag{11}$$

On the other hand, since E_i^2 $(i \geq 1)$ is an optimum Π-solution when the graph is weighted by w_i, $w_i(E_i^2) \geq w_i(E_i^*)$. Hence:

$$val(S^2) = \sum_{i=1}^r p_i w_i(E_i^2) \geq \sum_{i=1}^r p_i w_i(E_i^*) \tag{12}$$

Finally, since the algorithm outputs the better of the two $\tilde{\Pi}$-solutions S^1 and S^2, we get, by adding equations (11) and (12) with coefficients 1 and $r-1$:

$$(2r-1)val(S) \geq r \times w_0(E_0^*) + r \times \sum_{i=1}^r p_i w_i(E_i^*) = r \times val(S^*). \tag{13}$$

The result follows. □

It can be easily shown that the ratio given in Theorem 6 is tight. We show a tight example in the case of 2-stage max Mat.

Consider the instance on $r+3$ vertices u, v, w and t_i, $i = 1, \cdots, r$, and the following $r+1$ edges:

- An edge $[u, v]$ with weight $r-1$ in the first stage and weight 0 in all second stage scenarios;
- r edges $[w, t_i]$, $i = 1, \cdots, r$, with weight $1+\varepsilon$ ($\varepsilon > 0$) in the first stage, r in the scenario i of the second stage, and 0 in any other second stage scenario.

The scenario probabilities are $\frac{1}{r}$. The optimal $\tilde{\Pi}$-solution is clearly $(\{[u, v]\}, \{[w, t_1]\}, \ldots, \{[w, t_r]\})$. Its value is $r - 1 + \sum_{s=1}^{r} \frac{1}{r} r = 2r - 1$. In addition, the $\tilde{\Pi}$-solution returned by the approximation algorithm is $(\{[u, v], [w, t_i]\}, \emptyset, \ldots, \emptyset)$ (for some t_i). Its value is $r - 1 + 1 + \varepsilon = r + \varepsilon$. The approximation ratio is therefore $\frac{r+\varepsilon}{2r-1}$, which tends to $\frac{r}{2r-1}$ when ε tends to zero.

Note that the following refinement of Algorithm `2StageApx` does not achieve a better approximation ratio: after determining the edges chosen in the first stage in the globally-feasible $\tilde{\Pi}$-solution, determine the edges chosen in the second stage by optimizing separately for each scenario i in the remaining partial subgraph valued by $p_i w_i(e)$. Indeed, on the previous instance, no additional edge can be chosen after the ones of the first stage have been selected.

Theorem 7 *For 2-stage max Mat, `2StageApx` is a $\frac{\Delta+1}{2\Delta+1}$-approximation algorithm, where Δ is the degree of the graph. Moreover, if the graph is bipartite, then it is a $\frac{\Delta}{2\Delta-1}$-approximation algorithm.*

PROOF. The result is based on the following well known property: If G has a maximum degree Δ, then its edges can be $(\Delta+1)$-colored, *i.e.* partitioned into $(\Delta+1)$ matchings $N_1, \ldots, N_{\Delta+1}$ within polynomial time (it is known as the Vizing's theorem [14]).

Consider as previously an optimum solution $S^* = (E_0^*, \ldots, E_r^*)$. For any $i = 1, \ldots, \Delta+1$, we consider the subset N_i^* of N_i of edges used by S^* in the second stage (formally, $N_i^* = N_i \cap (\cup_{i=1}^{r} E_i^*)$). Each of these edges belong to some of the second-stage scenarios for S^*, hence its contribution in $val(S^*)$ is at most $\sum_{i=1}^{r} p_i w_i(e)$. In particular, since $w^1(e) = \max\{w_0(e), \sum_{i=1}^{p} p_i w_i(e)\} \geq \sum_{i=1}^{p} p_i w_i(e)$, we have:

$$\sum_{i=1}^{r} p_i w_i(E_i^*) \leq \sum_{i=1}^{\Delta+1} w^1(N_i^*).$$

Now, remark that $E_0^* \cup N_i^*$ is a matching, since on the one hand both E_0^* and N_i^* are matchings, and on the other hand any edge of N_i^* has been chosen in the second stage by S^* hence cannot

be adjacent to an edge of E_0^*. Since E^1 is a maximum weight matching in G weighted by w^1, we get:

$$val(S^1) = w^1(E^1) \geq w^1(E_0^*) + w^1(N_i^*) \tag{14}$$

Making the sum of equation (14) for $i = 1, \ldots, \Delta + 1$, we obtain:

$$\begin{aligned}(\Delta+1)val(S^1) &\geq (\Delta+1)w^1(E_0^*) + \sum_{i=1}^{\Delta+1} w^1(N_i^*) \\ &\geq (\Delta+1)w_0(E_0^*) + \sum_{i=1}^{r} p_i w_i(E_i^*)\end{aligned} \tag{15}$$

Since the algorithm outputs the better of the two solutions S^1 and S^2, we get, by adding equations (15) and (12) (of course still valid) with coefficients 1 and Δ:

$$(2\Delta+1)apx(I) \geq (\Delta+1)w_0(E_0^*) + (\Delta+1)\sum_{i=1}^{r} p_i w_i(E_i^*) = (\Delta+1)val(S^*). \tag{16}$$

The result follows.

The case of bipartite graphs is completely similar, up to the fact that in this case the edges of the graph can be partitioned into Δ matchings (it is known as the König's theorem [11]). This improvement leads to the inequality $(2\Delta - 1)apx(I) \geq \Delta val(S^*)$. □

Consider the example showing the tightness of Theorem 6. The graph is bipartite and has degree $\Delta = r$. The approximation ratio is $\frac{r+\varepsilon}{2r-1} = \frac{\Delta+\varepsilon}{2\Delta-1}$. This shows the tightness of the bound for bipartite graphs in Theorem 7.

Concerning bound in general graphs, we now show the tightness for any even Δ [1] . We consider a graph G on $\Delta + 3$ vertices constituted by two vertices u and v linked by an edge, and a disjoint complete graph $K_{\Delta+1}$ on $\Delta + 1$ vertices. It is well known that the edges of the clique $K_{\Delta+1}$ can be decomposed in $\Delta + 1$ matchings $N_1, \cdots, N_{\Delta+1}$, each of size $\Delta/2$. We consider $r = \Delta + 1$ scenarios in the second stage, each having a probability $1/r = 1/(\Delta + 1)$, and the following weights:

- the edge $[u, v]$ has weight $\Delta^2/2$ in the first stage, and 0 in any scenario of the second stage.
- each edge e in N_i has weight $1 + \varepsilon$ in the first stage, weight $\Delta + 1$ in the i^{th} scenario of the stage, and weight 0 in all other second stage scenarios.

Since $K_{\Delta+1}$ has an odd number of vertices, any maximum matching in $K_{\Delta+1}$ contains $\Delta/2$ edges. Since in the first step of the algorithm, each edge of the clique receives weight $1 + \varepsilon$,

[1] Note that the lower bound in bipartite graph, valid for any Δ, is already an almost tight bound for general graphs.

it means that the solution S^1 computed by the algorithm has value $\Delta^2/2 + (1+\varepsilon)\Delta/2 = \Delta(\Delta+1)/2 + \varepsilon\Delta/2$.

On the other hand, by construction, each matching N_i is a matching of maximum weight in the i^{th} scenario and has a cost $|N_i|(\Delta+1) = (\Delta+1)\Delta/2$. This implies that the second solution computed by the algorithm has value $(\Delta+1)\Delta/2$.

Now, an optimum solution is given by taking $[u,v]$ in the first stage, and N_i in the i^{th} scenario in the second stage. It has value $\Delta^2/2 + (\Delta+1)\Delta/2 = \Delta(2\Delta+1)/2$. The approximation ratio tends to $(\Delta+1)/(2\Delta+1)$ when ε tends to 0.

6 Heuristic

We propose a heuristic `2StageHeur`(α) where $\alpha \in \mathbb{R}$ is a parameter such that $0 \leq \alpha \leq 1$. It is an attempt to overcome a weakness of `2StageApx`. Let us go back to the tight example of `2StageApx` given above. When the approximation algorithm is executed, an edge incident to w, say $(w, t_{i'})$ for some $i' \in \{1, \ldots, r\}$, is put into E_0^1. Then edge (w, t_i) cannot be put into E_i^1 at the second stage when scenario i occurs. This is due to the fact that $w_0(w, t_{i'}) = 1 + \varepsilon$ is slightly larger than $\sum_{i=1}^{r} p_i w_i(w, t_{i'}) = 1$. The difference is just by an ε but the value of the $\tilde{\Pi}$-solution returned by the approximation algorithm is asymptotically 1/2 times the optimum value.

In order to overcome this drawback, we propose to introduce two potential functions $\Phi_1 : V \rightarrow \mathbb{R}$ and $\Phi_2 : V \rightarrow \mathbb{R}$, one for each stage. Actually Φ_1 (resp. Φ_2) assigns a value to each node $v \in V$ which corresponds to the best first stage (resp. best second stage) contribution of edges incident to v to the value of the resulting $\tilde{\Pi}$-solution. For the tight example, $\Phi_1(w) = 1 + \varepsilon$ (when $[w, t_{i'}]$ is taken at the first stage) whereas $\Phi_2(w) = r$ (when $[w, t_i]$ is taken at each scenario i of the second stage). Here we see a gap between $\Phi_1(w)$ and $\Phi_2(w)$. Therefore it would be interesting to define a threshold α such that no first stage edge incident to a node $v \in V$ is allowed if $\Phi_1(v)/\Phi_2(v) < \alpha$.

The potential argument seems particularly relevant when the variation on the degree of a node in all feasible Π-solutions is small. This is the case for a matching (a node has degree 1 or 0). However this is *a priori* not the case for a spanning tree. Therefore we present our heuristic in a general form (i.e. for problem $\tilde{\Pi}$) but we only test it on the 2-stage max Mat problem.

The heuristic consists in returning the best solution among three $\tilde{\Pi}$-solutions $\{S^1, S^2, S^3\}$ where S^1 and S^2 are those already computed by `2StageApx` and the third one takes into account the potential function. Note that as a consequence the performance of `2StageHeur`(α) is at least as good as `2StageApx`. More formally, the heuristic `2StageHeur`(α) is as follows:

(1) $E' = \mathcal{A}_\Pi(G, w^1, \emptyset)$ where $w^1(e) = \max(w_0(e), \sum_{i=1}^{r} p_i w_i(e))$
(2) $E_0^1 = \{e \in E' : w_0(e) > \sum_{i=1}^{r} p_i\, w_i(e)\}$

(3) For $i = 1, \ldots, r$ let $E_i^1 = \mathcal{A}_\Pi(G, w_i, E_0^1) \setminus E_0^1$
(4) Let $S^1 = (E_0^1, E_1^1, \ldots, E_r^1)$
(5) For all $v \in V$, let $\Phi_1(v) = \sum_{[u,v] \in E_0^1} w_0\big([u,v]\big)$
$\setminus *$ we suppose that $\sum_{[u,v] \in E_0^1} w_0\big([u,v]\big) = 0$ if $\{[u,v] \in E_0^1\} = \emptyset$ $*\setminus$
(6) For $i = 1, \ldots, r$ let $E_i^2 = \mathcal{A}_\Pi(G, w_i, \emptyset)$
(7) Let $S^2 = (\emptyset, E_1^2, \ldots, E_r^2)$
(8) For all $v \in V$, let $\Phi_2(v) = \sum_{i=1}^r \sum_{[u,v] \in E_i^2} p_i \, w_s\big([u,v]\big)$
$\setminus *$ we suppose that $\sum_{(u,v) \in E_i^2} p_i \, w_s\big((u,v)\big) = 0$ if $\{(u,v) \in E_i^2\} = \emptyset$ $*\setminus$
(9) Let $E_0^3 = \{[u,v] \in E_0^1 : \big(\Phi_1(u) \geq \alpha \Phi_2(u)\big) \wedge \big(\Phi_1(v) \geq \alpha \Phi_2(v)\big)\}$.
(10) Let $S^3 = (E_0^3, E_1^3, \ldots, E_r^3)$
(11) For $i = 1, \ldots, r$ let $E_i^3 = \mathcal{A}_\Pi(G, w_i, E_0^3) \setminus E_0^3$
(12) Return the best $\tilde{\Pi}$-solution S between S^1, S^2 and S^3

As we will see in the next section, the heuristic, though not being proved better when considering the theoretical worst case performance ratio, will compute better solutions in practical experiments.

7 Numerical experiments

The various algorithms for 2-stage max Mat have been implemented in JAVA and the computational tests were carried out on a PC with a Pentium IV CPU 3.6Ghz processor and 3.5GB of RAM. Following Kong and Schaefer [10], our tests were performed on a set of randomly generated instances of the two-stage assignment problem (i.e., matching in bipartite graphs), with 10 vertices in each side of the bipartition. Each time a standard assignment problem required to be solved (in the approximation algorithms), we used ILOG CPLEX 11.100 to compute the solution. We compare the values of the $\tilde{\Pi}$-solutions found by the approximation algorithms with those of the optimal $\tilde{\Pi}$-solutions obtained by using a mixed integer programming formulation in CPLEX. We have studied two kind of instances: correlated ones where the weights of an edge in the various scenarios of the second stage are correlated (which appears to be quite natural), and the uncorrelated ones where the weights in the various scenarios are independent.

The weights of the edges for the first stage were randomly drawn using a gaussian distribution $N(10, 15)$ (i.e., a gaussian distribution of mean 10 and standard deviation 15). For the correlated instances, the weight of an edge e under scenario s in the second stage are set using the formula $x_e + Y_s$, where x_e is a value randomly drawn (once for each edge e) using a gaussian distribution $N(10, 15)$ and Y_s is a random variable following a gaussian distribution $N(0, 5)$. For the uncorrelated instances, the weights of the edges for both the first and the second stage were randomly drawn using a gaussian distribution $N(10, 15)$ (the random variables for the first stage and the various scenarios of the second stage are assumed to be independent). Note that, in all cases, weights are set to 0 when negative values are drawn.

We compare here the following approximation algorithms for 2-stage max Mat:

#sc	CPLEX	(1/2)-Apx		2StageApx		R-2StageApx	2StageHeur(0.8)
	% 1^{st} stage	deviation	ratio	deviation	ratio		
2	(0,48,90)	(0,42,70)	(0.807,0.905,1) 1/100	(0,4,20)	(0.985, 0.999, 1) 80/100	(0.99,1,1) 91/100	(0.99,1,1) 91/100
3	(0,48,90)	(0,46,80)	(0.803,0.9,1) 2/100	(0,6,30)	(0.974, 0.998, 1) 78/100	(0.974,0.999,1) 90/100	(0.974,0.999,1) 92/100
5	(10,46,90)	(10,43,70)	(0.765,0.894,0.992) 0/100	(0,7,30)	(0.976, 0.996, 1) 52/100	(0.984,0.999,1) 79/100	(0.985,0.999,1) 80/100
10	(10,47,90)	(10,45,70)	(0.805,0.904,0.981) 0/100	(0,8,50)	(0.974, 0.997, 1) 56/100	(0.99,0.999,1) 80/100	(0.99,0.999,1) 83/100
20	(10,45,100)	(0,41,70)	(0.816,0.904,1) 1/100	(0,8,50)	(0.972, 0.997, 1) 34/100	(0.987,0.999,1) 77/100	(0.987,0.999,1) 78/100

Table 3
Results on correlated instances.

- `(1/2)-Apx`: the $\frac{1}{2}$-approximation algorithm of Kong and Schaefer [10],
- `2StageApx`: the $\frac{r}{2r-1}$-approximation algorithm proposed in Section 5,
- `R-2StageApx`: the refined version of `2StageApx`, where, after determining the edges chosen in the first stage in the globally-feasible $\tilde{\Pi}$-solution, one determines the edges chosen in the second stage by optimizing separately for each scenario,
- `2StageHeur`(α): the heuristic algorithm described in Section 6.

Tables 3 and 4 present the practical approximation ratios obtained by the various algorithms on 100 random instances. Column $\#sc$ displays the number of scenarios. Column `CPLEX` displays the (min, average, max) percentage of edges chosen in the first stage in the optimal $\tilde{\Pi}$-solutions computed by CPLEX on the 100 random instances. Column `(1/2)-Apx` (resp. `2StageApx`) displays the deviation in the percentage of first stage edges in the $\tilde{\Pi}$-solution returned by `(1/2)-Apx` (resp. `2StageApx`) compared to the optimal solution computed by CPLEX. For all the approximation algorithms, the practical approximation ratios are indicated, as well as the number x of times an optimal $\tilde{\Pi}$-solution has been returned (value $x/100$ on the second line). One can observe that `2StageApx` outperforms `(1/2)-Apx` on correlated instances. This can be explained by the fact that, for such instances, it is interesting to mix edges taken in the first and second stage. `(1/2)-Apx` is not able to do that, contrarily to `2StageApx`. Hence, the structure (i.e., the proportions of first and second stage edges) of the $\tilde{\Pi}$-solution returned by `2StageApx` is closer to the structure of an optimal $\tilde{\Pi}$-solution than the one returned by `(1/2)-Apx`. This is illustrated by the results in columns 'deviation': for correlated instances, one can see that the deviation is much smaller for `2StageApx` than for `(1/2)-Apx`. Refined version `R-2StageApx` as well as heuristic version `2StageHeur`(0.8) make it possible to even slightly improve the obtained results. For uncorrelated instances, the conclusions are similar to those for correlated instances, but the differences are less pronounced.

#sc	CPLEX	(1/2)-Apx		2StageApx		R-2StageApx	2StageHeur(0.8)
	% 1^{st} stage	deviation	ratio	deviation	ratio		
2	(0,45,80)	(0,42,80)	(0.796,0.916,1) 2/100	(0,25,60)	(0.867, 0.956, 1) 4/100	(0.875,0.966,1) 9/100	(0.902,0.971,1) 9/100
3	(10,48,100)	(0,42,70)	(0.849,0.931,1) 2/100	(0,28,60)	(0.896, 0.959, 1) 6/100	(0.907,0.968,1) 11/100	(0.907,0.970,1) 11/100
5	(10,45,100)	(0,41,80)	(0.856,0.944,1) 5/100	(0,36,80)	(0.885, 0.957, 1) 6/100	(0.91,0.962,1) 7/100	(0.91,0.965,1) 8/100
10	(0,46,100)	(0,38,80)	(0.865,0.953,1) 7/100	(0,35,70)	(0.896, 0.961, 1) 13/100	(0.896,0.966,1) 14/100	(0.896,0.967,1) 14/100
20	(0,46,100)	(0,39,80)	(0.877,0.955,1) 4/100	(0,35,80)	(0.897, 0.962, 1) 9/100	(0.91,0.967,1) 10/100	(0.91,0.968,1) 10/100

Table 4
Results on uncorrelated instances.

8 Concluding remarks

In this article we studied the two-stage stochastic versions of two classical polynomial problems in combinatorial optimization: maximum weight matching and maximum weight spanning tree. Our contribution is computational complexity results, polynomial cases, a general approximation algorithm and an empirical analysis of the algorithm. Those results improve the one of Kong and Schaefer [10].

Some questions remain open, in particular the computational complexity of 2-Stage max Mat and 2-Stage max ST if the number of scenarios is a fixed constant (e.g. $r = 2$). It would also be interesting to close the gap between our negative results (**APX**-completeness) and the positive result ($r/(2r-1)$-approximation). We believe that an approximation algorithm with a ratio better than $r/(2r-1)$ can be built.

References

[1] J. Birge, F. Louveaux, Introduction to Stochastic Programming, Springer, 1997.

[2] A. D. Flaxman, A. M. Frieze, M. Krivelevich, On the random 2-stage minimum spanning tree, Random Struct. Algorithms 28 (1) (2006) 24–36.

[3] M. R. Garey, D. S. Johnson, L. J. Stockmeyer, Some simplified NP-complete graph problems, Theor. Comput. Sci. 1 (3) (1976) 237–267.

[4] M. Gondran, M. Minoux, Graphes et algorithmes, Eyrolles, Paris, 1979.

[5] A. Gupta, M. Pál, R. Ravi, A. Sinha, Boosted sampling: approximation algorithms for stochastic optimization, Proc. of the 36th Annual ACM Symposium on Theory of Computing (STOC'04) (2004) 417–426.

[6] A. Gupta, M. Pál, R. Ravi, A. Sinha, What about wednesday? approximation algorithms for multistage stochastic optimization, Proc. of APPROX-RANDOM'05, Springer LNCS 3624 (2005) 86–98.

[7] N. Immorlica, D. R. Karger, M. Minkoff, V. S. Mirrokni, On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems, Proc. of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2004) (2004) 691–700.

[8] P. Kall, S. Wallace, Stochastic Programming, John Wiley and Sons, 1994.

[9] I. Katriel, C. Kenyon-Mathieu, E. Upfal, Commitment under uncertainty: Two-stage stochastic matching problems, Proc. of ICALP'07, Springer LNCS 4596 (2007) 171–182.

[10] N. Kong, A. J. Schaefer, A factor 1/2 approximation algorithm for two-stage stochastic matching problems, European Journal of Operational Research 172 (3) (2006) 740–746.

[11] D. König, Über graphen und iher anwendung auf determinantentheorie und mengenlehre, Math. Ann. 77 (1916) 453–465.

[12] C. H. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes, J. Comput. Syst. Sci. 43 (3) (1991) 425–440.

[13] R. Ravi, A. Sinha, Hedging uncertainty: Approximation algorithms for stochastic optimization problems, Math. Program. 108 (1) (2006) 97–114.

[14] V. G. Vizing, On an estimate of the chromatic class of a p-graph, Diskret. Analiz. 3 (1964) 25–30.