On the enumeration of non-dominated matroids with imprecise weights

Tom Davot, Tuan-Anh Vu, Sébastien Destercke, David Savourey

On the enumeration of non-dominated matroids with imprecise weights*

Tom Davot[1,2], Tuan-Anh Vu[3], Sébastien Destercke[2], David Savourey[2]

[1] School of Computing Science, University of Glasgow, UK
[2] Université de Technologie de Compiègne, CNRS, Heudiasyc (Heuristics and Diagnosis of Complex Systems), CS 60319 - 60203 Compiègne Cedex, surname.name@hds.utc.fr
[3] Univ. Artois, UR 3926, Laboratoire de Genie Informatique et d'Automatique de l'Artois (LGI2A), Bethune, France, tanh.vu@univ-artois.fr

Abstract. Many works within robust combinatorial optimisation consider interval-valued costs or constraints. While most of these works focus on finding a unique solution following a robust criteria such as minimax, a few consider the problem of characterising a set of possibly optimal solutions. This paper is situated within this line of work, and considers the problem of exactly enumerating the set of possibly optimal matroids under interval-valued costs. We show in particular that each solution in this set can be obtained through a polynomial procedure, and provide an efficient algorithm to achieve the enumeration.

Keywords: Enumeration · Optimisation · Interval Costs · Matroids

1 Introduction

There is little question that solving combinatorial optimisation problems is both a useful and challenging task, be it to design networks, to optimise industrial processes, to establish a schedule, etc. It is also unquestionable that in such problems, uncertainties in the constraints or the objective functions can occur. This has given rise to various approaches to account for these uncertainties, with mainly the two frameworks that are stochastic optimisation, that uses probabilistic modelling, and robust optimisation, that uses (convex) sets of scenarios as uncertainty models, with other frameworks such as evidence theory trying to bridge the gap between the two.

This paper is mainly concerned with the case of robust optimisation, where our uncertainty about costs is given by intervals. Such a topic has attracted some attention in the past (one can check, for instance, the book [10] for a good reference on the topic). In such issues, two typical problems are considered:

– finding a unique robust solution, often in the form of a minimax solution under interval uncertainty (see, e.g., Yaman et al. [19] for the case of minimum

* Preprint version. The editor version can be found at https://doi.org/10.1016/j.ijar.2024.109266

spanning trees). Of course other criteria to find a robust unique solution can be adopted [2,15,9], such as the maximin regret, that has also been considered in the case of matroids [11];
– finding *possible* and *necessary* elements, where an element is possibly optimal if it is part of an optimal solution for at least one scenario, and necessarily optimal if it is part of all optimal solutions, irrespectively of the chosen scenario within the intervals. For example [19] investigate the concept of weak (possible) and strong (necessary) edges in interval-valued minimum spanning trees, that is, edges that belong to at least one possibly optimal solution and to every possibly optimal solution, respectively. The same problem has been considered for matroids [12], the combinatorial problems we will consider in this paper.

In this paper, our interest is closer, yet different, of the latter problem rather than the former. More precisely, we want to consider the set of all possibly optimal solutions, and to enumerate efficiently and exactly such solutions. Such a problem may be important if, e.g., one wants to browse the Pareto front of optimal solutions under uncertainty. Note that the problem of finding possibly and necessarily optimal elements is not equivalent to that, as determining possibly and necessarily optimal elements only provides a possibly rough outer-approximation, as not all feasible solutions composed of possibly optimal elements will be an optimal solution for one scenario. This is illustrated by the next simple example.

Example 1. Consider a connected graph $G = (V, E)$ with associated interval costs $[\underline{c}_{ij}, \overline{c}_{ij}]$ for the edge $\{i, j\}$, and the problem of finding a minimum spanning tree, i.e., a subset $T \subseteq E$ such that (V, T) forms a tree and has minimal cost. Figure 1 shows a graph with four nodes and five edges with their associated interval-valued costs. It can be checked that all edges can be in some optimal solutions, but that no edges is present in every optimal solution. So the set of trees described by the possibly optimal elements is simply the set of all possible trees one can construct.

Yet, one cannot find a scenario where the edges $\{1, 3\}$ and $\{2, 3\}$ are simultaneously present in the solution, as whatever the chosen scenario, if $\{1, 3\}$ is present, one can always swap the edge $\{2, 3\}$ for $\{1, 2\}$ and get a lower score, as the score of $\{1, 2\}$ will always have a lower score than $\{2, 3\}$. This means that the set of trees that are optimal for one scenario is in this case a strict subset of those obtained by combining possibly optimal elements.

In a previous conference paper [5], we looked at the problem of enumerating minimum spanning trees with interval-valued scores. This paper is an extension of this paper, as we look here at much more general (and abstract) combinatorial problems than minimum spanning trees, which are matroids. Matroids are very general structures that embed many standard problems:

– the problem of scheduling tasks of constant time on a single machine with different associated deadlines and penalties corresponds to a specific transversal matroid [13, Chapter 7, Section 4];

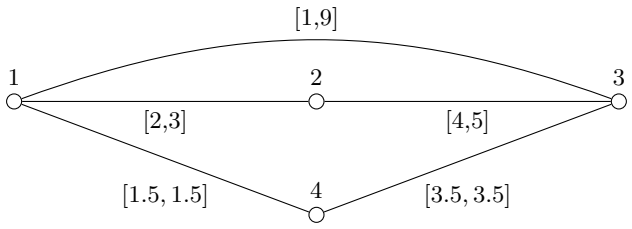

Fig. 1. Interval-valued graph of Example 1

- consider a set E of n jobs, where each completed job j brings a profit w_j, and a set A of m agents, each of them able to complete a possible subset of jobs. The task of assigning one job to each agent (that they can complete) while maximizing the overall profit is also a specific transversal matroid problem [7];
- the aforementioned problem of finding a minimum spanning tree corresponds to the specific subclass of graphic matroid.

Appendices A and B provide more details about such problems. For our motivation, it is sufficient to know that matroid problems can be commonly encountered.

Our paper is structured as follows: Section 1 introduces the notion of matroid and of minimum base, which is then made imprecise, allowing us to state the problem considered in this paper. Section 3 then describes some results that are necessary steps to demonstrate our main results. Our main results are then stated in Sections 4 and 5, where we derive conditions for an element being possible or necessary given a partially instantiated solution, who in turn allow us to provide an enumeration algorithm that has polynomial time delay, implying that enumeration can be done efficiently, even if its overall time still depends on the size of the set of possibly optimal solutions. Finally, Section 6 provides some experiments giving some first hints at situation where our enumeration methods are particularly efficient.

2 Notations and problem description

2.1 Matroid

Let $M = (E, \mathcal{B})$ be a pair constituted by a set of elements E and a collection $\mathcal{B}$ of subsets of E such that $B_1 \not\subseteq B_2$, for any two subsets $B_1 \in \mathcal{B}$ and $B_2 \in \mathcal{B}$. A subset $B \in \mathcal{B}$ is called a *base*. A subset $I \subseteq E$ is *independent* if there is a base $B \in \mathcal{B}$ such that $I \subseteq B$. Notice that in particular, $\varnothing$ is an independent set. A *circuit* is a minimal subset $C \subseteq E$ such that C is not independent. M is a *matroid* if it has the following property.

(1) Let I_1 and I_2 be two independent sets such that $|I_1| < |I_2|$. There is an element $e \in I_2 \setminus I_1$ such that $I_1 \cup \{e\}$ is an independent set.

In a matroid, every base has the same cardinality. Let $E' \subseteq E$ be a subset of elements. We denote $\mathcal{I}(E')$ (or simply $\mathcal{I}$ if $E' = E$) the set of independent sets included in E'. In other words, $I \in \mathcal{I}(E')$ if I is an independent set of M and $I \subseteq E'$, that is to the maximal cardinality of an independent set that is a subset of E'. Samewise, we denote $\mathcal{C}(E')$ (or simply $\mathcal{C}$ if $E = E'$) the set of circuits included in E'. In other words, $C \in \mathcal{C}(E')$ if C is a circuit of M and $C \subseteq E'$. The *rank function* is a function $r : \mathcal{P}(E) \to \mathbb{N}$ that associates every subset E' to the maximum cardinality of an independent set in $\mathcal{I}(E')$. Notice that for every base B, we have $|B| = r(E)$. For the sake of simplicity, we call $r(E)$ the rank of M. The *closure* of a subset E', denoted $cl(E')$ is a superset of E' containing every element that can be added to E' without increasing its rank. More formally, $cl(E') = \{e \in E \mid r(E' \cup \{e\}) = r(E')\}$. E' is said *closed* if $cl(E') = E'$. A *hyperplane* is a closed set of rank $r(E) - 1$.

In Example 2, we depict a matroid that will be used as an illustration throughout the rest of this article.

Example 2. Consider the matroid $M = (E, \mathcal{B})$ with $E = \{e_1, \ldots, e_7\}$ and

$$\mathcal{B} = \left\{ \begin{array}{l} \{e_1, e_2, e_3, e_5, e_6\}, \{e_1, e_2, e_3, e_5, e_7\}, \{e_1, e_2, e_3, e_6, e_7\}, \\ \{e_1, e_2, e_4, e_5, e_6\}, \{e_1, e_2, e_4, e_5, e_7\}, \{e_1, e_2, e_4, e_6, e_7\}, \{e_1, e_2, e_5, e_6, e_7\}, \\ \{e_1, e_3, e_4, e_5, e_6\}, \{e_1, e_3, e_4, e_5, e_7\}, \{e_1, e_3, e_4, e_6, e_7\}, \{e_1, e_3, e_5, e_6, e_7\}, \\ \{e_2, e_3, e_4, e_5, e_6\}, \{e_2, e_3, e_4, e_5, e_7\}, \{e_2, e_3, e_4, e_6, e_7\}, \{e_2, e_3, e_5, e_6, e_7\} \end{array} \right\}.$$

The matroid M has three circuits $C_1 = \{e_1, e_2, e_3, e_4\}$, $C_2 = \{e_4, e_5, e_6, e_7\}$ and $C_3 = E \setminus \{e_4\}$ and has rank 5.

In matroid theory, we often use matroid oracle. The most used is the *independence oracle* which, given a subset of elements E', returns `true` if E' is independent and *false* otherwise. We denote i_M the time complexity of the independence oracle for the matroid M. In this article, we also use a port oracle that was introduced by Coullard and Hellerstein [4]. A *port oracle* is an oracle that, given a subset of elements E' and an element $e' \in E'$, returns `true` if there is a circuit in $\mathcal{C}(E')$ that contains e' and `false` otherwise. We denote p_M the time complexity of the port oracle for the matroid M.

Remark 1. As noted in [4], the port oracle of M can be implemented by using the independence oracle of M in $\mathcal{O}(|E'| \cdot i_M)$ as follows: generate a maximal independent subset I of $E' \setminus \{e\}$ one element at a time, then test if $I \cup e$ contains a circuit.

Finally, we recall an important property of circuits, used later on.

Property 1 (Strong circuit elimination axiom). Let C_1 and C_2 be two distinct circuits and let $e_1 \in C_1 \cap C_2$ and $e_2 \in C_1 \setminus C_2$. There is a circuit $C_3 \subseteq (C_1 \cup C_2) \setminus e_1$ such that $e_2 \in C_3$.

2.2 Minimum base

A *weighted matroid* $M = (E, \mathcal{B}, W)$ is a matroid with a *weight function* $W : E \to \mathbb{R}$ that associates to each element e a weight $W(e)$. Given a subset $E' \subseteq E$, we denote $W(E')$ the sum of the weights of the elements of E', that is

$$W(E') = \sum_{e \in E'} W(e).$$

A base $B \in \mathcal{B}$ is *minimum* if $W(B)$ is minimum, that is, if there is no base $B' \in \mathcal{B}$ such that $W(B') < W(B)$. An important property of matroids is that it is possible to compute a minimum base using a greedy algorithm [16] (see Algorithm 1). Notice that we need to call an independence oracle in line 5, thus, the complexity of this algorithm is $\mathcal{O}(|E| \log |E| + |E| \cdot i_M)$.

Algorithm 1: Greedy Algorithm

```
   Data: A weighted matroid (E, B, W)
   Result: A minimum base B
1  sort E by increasing order of weight;
2  B ← ∅;
3  while E ≠ ∅ do
4  |  e ← first element in the ordered-list E ;
5  |  if B ∪ {e} is an independent set then
6  |  |  B ← B ∪ {e};
7  return B;
```

2.3 Imprecise weights and problem description

An *imprecise weight* $[\underline{\omega}, \overline{\omega}]$ is an interval of numbers. An *imprecise weighted matroid* $M = (E, \mathcal{B}, \Omega)$ is a matroid with a function Ω that associates to each element e an imprecise weight $[\underline{\omega}_e, \overline{\omega}_e]$. A *realization* $R : E \to \mathbb{R}$ of Ω is a weight function that associates to each element e a weight $w \in \Omega(e)$. We denote $\mathcal{R}_\Omega$ the set of realizations of Ω.

Let $E' \subseteq E$ be a subset of elements of E. Given a weight realization R, the weight of E', denoted $R(E')$ is the sum of the weights of its elements, that is, $R(E') = \Sigma_{e \in E'} R(e)$. Given two subsets of elements E_1 and E_2, we say that E_1 *dominates* E_2, denoted by $E_1 \succ E_2$ if,

$$\forall R \in \mathcal{R}_\Omega, R(E_1) < R(E_2).$$

Given two elements e_1 and e_2, we say that e_1 *dominates* e_2 if $\overline{\omega}_{e_1} < \underline{\omega}_{e_2}$. In the following, we are interested in the set of non-dominated bases.

$$\mathcal{B}[\Omega] := \{B \in \mathcal{B} \mid \nexists B' \in \mathcal{B}, B' \succ B\}. \tag{1}$$

An example of imprecise weighted matroid is depicted in Example 3. An element e is *possible* if there is a base $B \in \mathcal{B}[\Omega]$ such that $e \in B$. An element e is *necessary* if for every base $B \in \mathcal{B}[\Omega]$, we have $e \in B$. Kasperski et al. shown that it is possible to determine if an element is possible or necessary by using a greedy algorithm [12].

Theorem 1 ([12]). *Let $M = (E, \mathcal{B}, \Omega)$ be an imprecise weighted matroid, let $e \in E$ be an element and let ϵ be an infinitively small value.*

(a) Let $R_p \in \mathcal{R}_\Omega$ such that $R_p(e) = \underline{\omega}_e - \epsilon$ and $\forall e' \in E \setminus \{e\}, R_p(e') = \overline{\omega}_{e'}$. Let B be a minimum base under R_p, computed with a greedy algorithm. The element e is possible if and only if $e \in B$.
(b) Let $R_n \in \mathcal{R}_\Omega$ such that $R_n(e) = \overline{\omega}_e + \epsilon$ and $\forall e' \in E \setminus \{e\}$,$R_n(e')= \underline{\omega}_{e'}$. Let B be a minimum base under R_n, computed with a greedy algorithm. The element e is necessary if and only if $e \in B$.

In other words, an element e is possible (resp. necessary) if e belongs to a minimum base under the best (resp. worst) realization for e. The addition (resp. substraction) of ϵ is needed so that in case of a tie between e and another element in the greedy algorithm, e is considered first (resp. last). In this article, we address the problem of enumerating every base of $\mathcal{B}[\Omega]$.

Imprecise Minimum Base (IMB)
Input An imprecise weighted matroid $M = (E, \mathcal{B}, \Omega)$.
Output Enumeration of $\mathcal{B}[\Omega]$

Example 3. Consider the imprecise weighted matroid $M = (E, \mathcal{B}, \Omega)$ where E and $\mathcal{B}$ are described in Example 2 and,

$$\Omega = \left\{ \begin{array}{l} \Omega(e_1) = [1,4],\ \Omega(e_2) = [1,1],\ \Omega(e_3) = [1,1],\ \Omega(e_4) = [3,6], \\ \Omega(e_5) = [2,6],\ \Omega(e_6) = [5,7],\ \Omega(e_7) = [3,5] \end{array} \right\}.$$

The set of non-dominated bases is

$$\mathcal{B}[\Omega] = \left\{ \begin{array}{l} \{e_1, e_2, e_3, e_5, e_6\},\ \{e_1, e_2, e_3, e_5, e_7\},\ \{e_1, e_2, e_3, e_6, e_7\}, \\ \{e_2, e_3, e_4, e_5, e_6\},\ \{e_2, e_3, e_4, e_6, e_7\},\ \{e_2, e_3, e_5, e_6, e_7\} \end{array} \right\}.$$

Every element of M is possible and e_2 and e_3 are necessary. However, not all subsets of size 5 including $\{e_2, e_3\}$ are in $\mathcal{B}[\Omega]$, as for instance $\{e_1, e_2, e_3, e_4, e_5\}$ is not. The reason is similar to the one advocated in Example 1.

2.4 Partial solution

Let $M = (E, \mathcal{B}, \Omega)$ be an imprecise weighted matroid for which we want to enumerate every non-dominated base. A *partial solution* S is a pair of sets of elements $in(S)$ and $out(S)$ such that there is a non-dominated base $B \in \mathcal{B}[\Omega]$ that contains every element of $in(S)$ and no element of $out(S)$. In that case,

we say that B *is associated* to S. We denote $\mathcal{B}_S[\Omega]$ the set of bases of $\mathcal{B}[\Omega]$ associated to S. Formally,

$$\mathcal{B}_S[\Omega] = \{B \in \mathcal{B} \mid in(S) \subseteq B \wedge out(S) \cap B = \varnothing\}.$$

We denote $S_\varnothing$ the *empty partial solution* for which $in(S_\varnothing) = out(S_\varnothing) = \varnothing$. Notice that $\mathcal{B}[\Omega] = \mathcal{B}_{S_\varnothing}[\Omega]$. An example of partial solution is depicted in Example 4. Notice that for a partial solution S, we necessarily have $\mathcal{B}_S[\Omega] \neq \varnothing$. Hence, every disjoint pair of elements sets is not a partial solution since at least one non-dominated base needs to be associated to this pair (see Example 5). Let S be a partial solution. We extend the notion of possible and necessary elements for partial solutions as follows. An element $e \notin in(S) \cup out(S)$ is *possible* with respect to S if there is a base $B \in \mathcal{B}_S[\Omega]$ such that $e \in B$. Similarly, e is *necessary* with respect to S if for all bases $B \in \mathcal{B}_S[\Omega]$, we have $e \in B$. Notice that an element e is possible (resp. necessary) if and only if e is possible (resp. necessary) with respect to $S_\varnothing$.

Example 4. Pursuing Example 3, consider the partial solution S with $in(S) = \{e_7\}$ and $out(S) = \{e_1\}$. The set of non-dominated bases associated to S is

$$\mathcal{B}_S[\Omega] = \{\{e_2, e_3, e_4, e_6, e_7\}, \{e_2, e_3, e_5, e_6, e_7\}\}.$$

The elements e_3 and e_4 are necessary with respect to S and every element in $E \setminus out(S) \cup in(S)$ is possible with respect to S.

Example 5. Consider the pair of elements sets $in(S') = \{e_5\}$ and $out(S') = \{e_1, e_4\}$. S' is not a partial solution for the imprecise weighted matroid depicted in Example 3. Indeed, $B = \{e_2, e_3, e_5, e_6, e_7\}$ is the only base such that $in(S') \subseteq B$ and $out(S') \cap B \neq \varnothing$ and B does not belong to $\mathcal{B}$.

An important remark is that we cannot reuse Theorem 1 to determine if an element is possible/necessary with respect to some partial solution S, otherwise the problem of enumerating elements of $\mathcal{B}[\Omega]$ would be trivial. For example, consider the partial solution S given by Example 4: the element e_4 is necessary with respect to S. However, if we consider the realization R_n for which $R(e_4) = 6 + \epsilon$ and $R(e) = \underline{\omega}_e$ for every element $e \neq e_4$, then the greedy algorithm returns $B = \{e_2, e_3, e_5, e_6, e_7\}\}$ as a minimum base of $E \setminus out(S)$ which does not belong to $\mathcal{B}_S[\Omega]$.

In the algorithm developed in Section 5, we add the element in the partial solution by increasing order of the lower bound. Hence, we want to ensure that if any element e belongs to $in(S)$, then any element d with a smaller value $\underline{\omega}_d$ belongs to S. Therefore, we introduce the concept of nice partial solution as follows.

Definition 1. *A partial solution S is* nice *if for any pair of elements e_1 and e_2 such that $\underline{\omega}_{e_1} < \underline{\omega}_{e_2}$, we have $e_2 \in in(S) \cup out(S) \Rightarrow e_1 \in in(S) \cup out(S)$.*

Given a nice partial solution, an element $e \notin in(S) \cup out(S)$ is a *first non-assignated element* of S if e is a minimum among the elements that do not belong to S, that is, $e = \arg\min\{\underline{\omega}_e \mid e \in E \setminus (in(S) \cup out(S))\}$. An example of a nice partial solution is depicted in Example 6. Notice that the partial solution given in Example 4 is not a nice partial solution since $\underline{\omega}_{e_2} < \underline{\omega}_{e_7}$, $e_2 \notin in(S) \cup out(S)$ and $e_7 \in in(S)$.

Example 6. Consider the pair of elements sets $in(S) = \{e_1, e_3\}$ and $out(S) = \{e_2, e_5\}$. S is a nice partial solution. The two elements e_4 and e_7 are the two first non-assignated elements of S.

3 Preliminary results

In this section, we introduce some definitions and some preliminary results, that will prove useful in the rest of the paper.

3.1 Domination

To determine if a base B_1 dominates another base B_2 for an imprecise weighted matroid, we can use the following lemma.

Lemma 1. *Let $M = (E, \mathcal{B}, \Omega)$ be an imprecise weigthed matroid. Let $B_1 \in \mathcal{B}$ and $B_2 \in \mathcal{B}$ be two bases. B_1 dominates B_2 if and only if $B_1 \setminus B_2$ dominates $B_2 \setminus B_1$.*

Proof. Let R be any realization. We can establish the following relations.

$$\begin{gathered} R(B_1) < R(B_2) \\ \Leftrightarrow \\ R(B_1 \setminus B_2) + R(B_1 \cap B_2) < R(B_2 \setminus B_1) + R(B_1 \cap B_2) \\ \Leftrightarrow \\ R(B_1 \setminus B_2) < R(B_2 \setminus B_1). \end{gathered}$$

Thus, B_1 dominates B_2 if and only if $B_1 \setminus B_2$ dominates $B_2 \setminus B_1$. □

Corollary 1. *Let $M = (E, \mathcal{B}, \Omega)$ be an imprecise weighted matroid. Let $B_1 \in \mathcal{B}$ and $B_2 \in \mathcal{B}$ be two bases. B_1 dominates B_2 if and only if*

$$\sum_{e \in B_1 \setminus B_2} \overline{\omega}_e < \sum_{e \in B_2 \setminus B_1} \underline{\omega}_e.$$

Proof. Let R be a realization such that $R(e) = \overline{\omega}_e$ if $e \in B_1$ and $R(e) = \underline{\omega}_e$ otherwise. We have $R(B_1 \setminus B_2) = \sum_{e \in B_1 \setminus B_2} \overline{\omega}_e$ and $R(B_2 \setminus B_2) = \sum_{e \in B_2 \setminus B_1} \underline{\omega}_e$. Hence, if the property is false then we have $R(B_1 \setminus B_2) \geq R(B_2 \setminus B_1)$ and thus, B_1 does not dominate B_2. Further, for any realization R', we have $R'(B_1 \setminus B_2) \leq R(B_1 \setminus B_2)$ and $R'(B_2 \setminus B_1) \geq R(B_2 \setminus B_1)$. Hence, if $R(B_1 \setminus B_2) < R(B_2 \setminus B_1)$, then for any realization R', we have $R'(B_1 \setminus B_2) < R'(B_2 \setminus B_2)$ and thus, by Lemma 1, B_1 dominates B_2. □

3.2 Cocircuit

We now introduce the notion of cocircuit which is a central notion in our article. A cocircuit $X \subseteq E$ of a matroid M is usually defined as a circuit in the dual of M. For the purpose of this article, we will use the following definition.

Definition 2. *Let $M = (E, \mathcal{B})$ be a matroid and let I be an independent set of rank $r(E) - 1$. The set of elements $X = \{e \in E \mid I \cup \{e\} \in \mathcal{B}\}$ is a* cocircuit. *In other words, X contains any element that can be added to I without creating a circuit.*

Note that in Definition 2, $cl(I)$ is a hyperplane and the cocircuit X is nothing but the complement of $cl(I)$, i.e., $X = E \setminus cl(I)$. In fact, any cocircuit is the complement of a hyperplane.

Example 7. Consider the imprecise weighted matroid $M = (E, \mathcal{B})$ described in Example 2. The subset $X = \{e_1, e_4, e_6\}$ is a cocircuit of M. Indeed the only independent set of $rank(E) - 1$ in $E \setminus X$ is $I = \{e_2, e_3, e_5, e_7\}$ and for any element $e \in X$, $I \cup \{e\}$ is a base. However, the subset $X' = \{e_5, e_6, e_7\}$ is not a cocircuit: $I' = \{e_1, e_2, e_3\}$ is a maximal independent set in $E \setminus X'$ and $I' \cup \{e_5\}$ is not a base.

Notice also that if M is a matroid such that elements of E are the edges of a graph G and the bases are the spanning trees of G, then a cocircuit corresponds to a minimal cut-set of G, that is, a minimal subset of edges such that their deletion disconnect the graph[4]. The following property indicates that if a cocircuit X intersects a circuit C, then there are at least two elements of C in X, or in other words that a cocircuit can not intersect a circuit only once.

Property 2 (Orthogonality property). Let X be a cocircuit of a matroid $M = (E, \mathcal{B})$ and let $e \in X$. Let $C \in \mathcal{C}$ be a circuit containing e. We have $|X \cap C| \geq 2$.

We are going to introduce a result (Lemma 3) aiming to construct a specific cocircuit containing some specific elements and excluding some others. For this purpose, we need a preliminary lemma.

Lemma 2. *Let $A \subseteq E$ and let $E' \subseteq E$ such that*

(a) $A \cap E' = \varnothing$,
(b) for any subset $E'' \subset E'$, there is no circuit in $\mathcal{C}(A \cup E'')$ containing E'', and
(c) there is a circuit $C \in \mathcal{C}(A \cup E')$ containing E'.

Let I be a maximal independent set in A, there is a circuit D such that $E' \subseteq D$ and $D \setminus E' \subseteq I$.

[4] The kind of structured we considered in [5]

Proof. Let $e \in E'$. We construct a circuit D by induction and we ensure that at each step we have $e \in D$. First, we set $D := C$. By definition of C, we have $e \in D$. Then, we exhaustively apply the following rule. Let $x \in D \setminus I \cup E'$. Since I is maximal, there is a circuit C_x such that $x \in C_x$ and $C_x \setminus \{x\} \subseteq I$. By Property 1, there is a circuit $C' \subseteq D \cup C_x \setminus \{x\}$ containing e. We set $D := C'$. When the rule does not apply anymore, we obtain a circuit D such that $D \setminus E' \subseteq I$. Let $E'' = D \cap E'$. We have $E'' = E'$ since otherwise we would have $D \subseteq I \cup E''$, contradicting condition (b). Hence, we obtain a circuit D as thought. □

We are ready to state Lemma 3.

Lemma 3. *Let $M = (E, \mathcal{B}, \Omega)$ be an imprecise weighted matroid. Let $A \subseteq E$ and $B \subseteq E$ be two disjoint subsets of elements such that*

- *(i) for all element $a \in A$, there is no circuit containing a in $B \cup \{a\}$, and*
- *(ii) for all pair of elements $a_1, a_2 \in A$, there is a circuit containing a_1 and a_2 in $B \cup \{a_1, a_2\}$.*

There is a cocircuit X such that $X \cap B = \varnothing$ and $A \subseteq X$.

Proof. Let I_{init} be a maximal independent set in B. We construct an independent set I as follows. We initiate I by taking $I := I_{init}$, then we apply exhaustively the following rule: let e be an element of M, if $I \cup \{e\}$ does not contain a circuit and for any element $a \in A$, $I \cup \{e, a\}$ does not contain a circuit, then we set $I := I \cup \{e\}$ in I. Let X be the set of elements x such that $I \cup \{x\}$ is an independent set. Notice that I is a maximal independent set in $E \setminus X$. By construction, A is a subset of X. We show that X is a cocircuit of I. Toward a contradiction, suppose not, that is, it exists two elements $x_1 \in X$ and $x_2 \in X$ such that $I \cup \{x_1, x_2\}$ is an independent set. Consider the following cases.

- Suppose $x_1 \in A$ and $x_2 \in A$. By Lemma 2, there is a circuit D such that $D \setminus \{x_1, x_2\} \subseteq I$ contradicting $I \cup \{x_1, x_2\}$ being an independent set.
- Suppose $x_1 \in A$ and $x_2 \notin A$. By construction of I, x_2 does not belong to I because there is an element $a \in A$ such that there is a circuit C_2 in $I \cup \{x_2, a\}$ containing x_2 and a. Since a and x_2 belong to X, there is no circuit in $I \cup \{a\}$ nor in $I \cup \{x_2\}$. Hence, by Lemma 2, there is a circuit D_2 such that $a \in D_2$ and $D_2 \setminus \{a, x_2\} \subseteq I$ If $a = x_1$, then $I \cup \{x_1, x_2\}$ contains the circuit D_2 which is a contradiction. Suppose that $a \neq x_1$. By Lemma 2, there is a circuit D_1 such that $a \in D_1$ and $D_1 \setminus \{a, x_1\} \subseteq I$. Since $a \in D_1 \cup D_2$, by Property 1, there is a circuit D_3 such that $D_3 \subseteq D_1 \cup D_2 - a$. But then, $D_3 \setminus \{x_1, x_2\} \subseteq I$ and thus, $I \cup \{x_1, x_2\}$ contains the circuit D_3 which is a contradiction.
- Suppose $x_1 \notin A$ and $x_2 \notin A$. For each $i \in \{1, 2\}$, by construction of I, there is an element a_i such that there is a circuit C_i in $I \cup \{a_i, x_i\}$. By Lemma 2, there is a circuit D_i such that $\{a_i, x_i\} \subseteq D_i$ and $D_i \setminus \{a_i, x_i\} \subseteq I$. Moreover, by definition of A, there is a circuit containing a_1 and a_2. By Lemma 2, there is a circuit D_3 such that $\{a_1, a_2\} \subseteq D_3$ and $D_3 \setminus \{a_1, a_2\} \subseteq I$. We have $x_1 \notin D_1 \cap D_3$ and $a_1 \in D_1 \cap D_3$. So, by Property 1, there is a circuit D_4 such that $x_1 \in D_4$ and $D_4 \subseteq D_1 \cup D_3 - a_1$. Moreover, $D_4 \setminus \{x_1, a_2\} \subseteq I$.

If $a_2 \notin D_4$, then $D_4 - x_1 \subseteq I$ and thus, $I \cup \{x_1\}$ contains the circuit D_4 which is a contradiction. Suppose $a_2 \in D_4$. We have $x_1 \notin D_2 \cap D_4$ and $a_2 \in D_2 \cap D_4$. So, by Property 1, there is circuit D_5 such that $x_1 \in D_5$ and $D_5 \subseteq D_2 \cup D_5 - a_2$. Moreover, $D_5 \setminus \{x_1, x_2\} \subseteq I$. Thus, $I \cup \{x_1, x_2\}$ contains the circuit D_5 which is a contradiction.

Hence, it is not possible to add two elements of X in I without creating a circuit, and so, we obtain a cocircuit X as thought. □

Let B be a base and let X be a cocircuit. Let $e \in X \setminus B$. Let C be a circuit in $B \cup \{e\}$. We call every element e' in $C \cap B \cap X$ a *X-blocking element* for e in B (e' necessarily exists by Property 2). Notice that the subset $B' = B \cup \{e\} \setminus \{e'\}$ is also a base. We say that B' is *obtained by swapping* e and e' in B. We recall the local optimality property.

Property 3 (local optimality property [16]). Let B be a base of a weighted matroid $M = (E, \mathcal{B}, W)$. If B is not minimum, then there is another base B' such that $W(B') < W(B)$ and B' is obtained by swapping two elements $e \in B$ and $e' \notin B$. Notice that e and e' belong to the cocircuit induced by e and B.

Given a base B and an element $e \in B$, we sometimes need to designate the cocircuit containing every element that can be added in $B \setminus \{e\}$ without creating a circuit. For that, we introduce the following definition

Definition 3. *Let $M = (E, \mathcal{B})$ be a matroid, let B a base and let $e \in B$. Let $X = \{e' \in E \mid (B \setminus \{e\}) \cup \{e'\} \in \mathcal{B}\}$ be the cocircuit containing every element that can be added in $B \setminus \{e\}$ without creating a circuit. The cocircuit X is designated as the cocircuit* induced by e and B.

Notice that a cocircuit X induced by a base B and an element e contains every element e' such that there is a circuit $C \subseteq B \cup \{e'\}$ with $\{e, e'\} \subseteq C$. Also, notice that e is X-blocking for any element e' of X. Moreover, if there is another cocircuit X' such that e is X'-blocking for an element e' in B, then e' necessarily belongs to X.

Lemma 4. *Let B_1 be a base and, for all $1 \leq i \leq 2$ let $e_i \in B_1$ and X_i be the cocircuit induced by B_1 and e_i. Let e_3 be an element of $X_1 \setminus B_1$. Let B_2 be the base obtained by swapping e_1 and e_3 in B_1 and let Y_3 be the cocircuit induced by B_2 and e_2. We have*

$$(X_2 \cup X_3) \setminus (X_2 \cap X_3) \subseteq Y_3 \subseteq X_2 \cup X_3.$$

Proof. Let $e \notin B_1$ be an element such that $e \notin X_1 \cup X_2$. There is a circuit C in $B_1 \cup \{e\}$ that does not contain e_2 nor e_3. Hence, we have $(C \setminus \{e\}) \subseteq B_1 \setminus \{e_2\} = B_2 \setminus \{e_1\}$ and so, e does not belong to Y_3.

Let $e \in X_2 \setminus X_3$. If $e = e_2$, then $e \in Y_3$. Otherwise, there is a circuit C in $B_1 \cup \{e\}$ that contains e_2 and does not contain e_3. Hence, we have $C \setminus \{e\} \subseteq B_2$ and so, $e \in Y_3$.

Let $e \in X_3 \setminus X_2$. If $e = e_3$, then there is a circuit C_1 that contains e_1, e_3 and e_3 in $B_1 \cup \{e_1\}$. Hence, we have $C_1 \setminus \{e_3\} \subseteq B_2$ and so, $e \in Y_3$. Otherwise, there is a circuit C_2 in $B_1 \cup \{e\}$ that contains e_3 and does not contain e_2. Since $e_3 \in C_1 \cap C_2$, by Property 1, there is a circuit C_3 that contains e_2 and does not contains e_3 in $C_1 \cup C_2$. Moreover, C_3 contains e since otherwse we would have $C_3 \in B_2$, contradicting B_2 being a base. Hence, we have $C_3 \setminus \{e\} \subseteq B_2$ and so, $e \in Y_3$. □

3.3 Core

We now introduce the notion of the core of a cocircuit and show that an element is possible/necessary with respect to a partial solution if it respects some property related to a core of a cocircuit. This will be essential to derive algorithms incrementally building possibly optimal solutions.

Definition 4. *Let $M = (E, \mathcal{B}, \Omega)$ be an imprecise weighted matroid and let X be a cocircuit of M. The* core *of X is a subset $K_X \subseteq X$ such that there is no element of X that dominates K_X.*

Let X be a cocircuit, an element e is a *minimum element of X* if $\underline{\omega}_e$ is minimum in X, that is, $e = \arg\min\{\underline{w}_{e'} \mid e' \in X\}$. Notice that e dominates every element e' in $X \setminus K_X$. An example of a cocircuit with its core is depicted in Example 8. The next property show that the core of a cocircuit intersects any non-dominated base.

Property 4. Let $M = (E, \mathcal{B}, \Omega)$ be an imprecise weigthed matroid and let B be a base. B is non-dominated if and only if for every cocircuit (induced by an element) X, we have $K_X \cap B \neq \varnothing$.

Proof. Let B be a non-dominated base. Toward a contradiction, suppose there is a cocircuit X such that $K_X \cap B = \varnothing$. Let e be a minimum element of X. Let C be a the circuit in $B \cup \{e\}$. Let e' be an X-blocking element for e in B. Since $e' \notin K_X$, e' is dominated by e. Thus, by Lemma 1, the base obtained by swapping e' and e in B dominates B, contradicting B being not dominated.

For the other direction, assume that B is dominated. From Corollary 1, B is not optimal under a specific weight realization R where $R(e) = \underline{\omega}_e \ \forall e \in B$ and $R(e) = \overline{\omega}_e \ \forall e \notin B$. By Property 3, there is a base B' such that $B \setminus B' = \{e\}$, $B' \setminus B = \{e'\}$, and $R(B') < R(B)$. It follows that e' dominates e. Moreover, both e, e' are in the cocircuit X induced by B and e. Therefore, $K_X \cap B = \varnothing$. □

Example 8. Consider the imprecise weighted matroid $M = (E, \mathcal{B}, \Omega)$ described in Examples 2 and 3. Let $X = \{e_1, e_4, e_6\}$ be the cocircuit described in Example 7. The element e_1 is a minimum element of X and e_6 is dominated by e_1, thus, we have $K_X = \{e_1, e_4\}$. Consider the base $B = \{e_1, e_2, e_3, e_5, e_6\}$. The element e_1 is X-blocking for e_4 in B since there is the circuit $C_1 = \{e_1, e_2, e_3, e_4\}$ in $B \cup \{e_4\}$. Finally, the base $B' = \{e_2, e_3, e_4, e_5, e_6\}$ is obtained by swapping e_1 and e_4 in B.

Next lemma shows that if there is an element e that does not belong to a non-dominated base B_1, then if e belongs to a core of any cocircuit, then we can construct another non-dominated base by swapping e with another element in B_1.

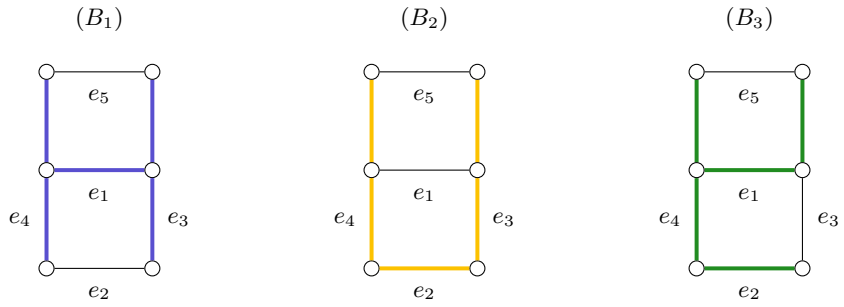

Fig. 2. Example of application of Lemma 5 in minimum spanning trees. The element e_2 belongs to the core of the cocircuit $X_1 = \{e_1, e_2, e_5\}$. The element e_3 has the maximum value $\underline{\omega}_{e_3}$ in the circuit $\{e_1, \ldots, e_4\}$. The bases B_2 and B_3 are obtained by swapping e_2 with respectively e_1 and e_3 in B_1. By Lemma 5(a), the base B_3 is non-dominated. Let $X_2 = \{e_2, e_3, e_5\}$ be the cocircuit induced by e_3 in B_2. If $e_3 \in K_{X_2}$, then By Lemma 5(b), B_2 is non-dominated.

Next lemma shows that it is possible to construct a non-dominated base from another by just swapping two elements of a same core. This gives us a nice way to create a new undominated base from an existing one.

Lemma 5. *Let B_1 be a non-dominated base and let $e_1 \in B_1$. Let X_1 be the cocircuit induced by e_1 and B_1. Let $e_2 \notin B_1$ be an element that belongs to K_{X_1}. Let C be the circuit in $B_1 \cup \{e_2\}$ and let e_3 be any element such that $e_3 = \arg\max\{\underline{\omega}_e \mid e \in C\}$ (possibly $e_2 = e_3$). Let B_2 be the base obtained by swapping e_2 and e_1 in B_1 and let B_3 be the base obtained by swapping e_2 and e_3 in B_1. We have the two following properties.*

(a) B_3 is non-dominated.
(b) If $e_3 \neq e_1$, let X_2 be the cocircuit induced by e_3 and the base B_2. If $e_3 \in K_{X_2}$, then B_2 is non-dominated.

Proof. (a) First, notice that if $e_2 = e_3$, then $B_1 = B_3$ and B_3 is trivially non-dominated. Thus, suppose $e_2 \neq e_3$. Toward a contradiction, suppose B_3 is dominated by another base. By Property 4, there is a cocircuit Y_3 such that $K_{Y_3} \cap B_3 = \varnothing$. We have $B_1 \cap K_{Y_3} = \{e_3\}$ since otherwise, we would have $B_1 \cap K_{Y_3} = \varnothing$ contradicting $B_1 \in \mathcal{B}[\Omega]$. By Property 2, there is an element $e_4 \in C \cap Y_3 \setminus \{e_3\}$. Moreover, by definition of e_3, we have $\underline{\omega}_{e_4} \leq \underline{\omega}_{e_3}$ thus, $e_4 \in K_{Y_3}$ which contradicts $B_3 \cap K_{Y_3} = \varnothing$. We can conclude that B_3 is not dominated.

(b) Toward a contradiction, suppose B_2 is dominated by another base.
By Property 4, there is a cocircuit Y_3 induced by an element $e_4 \in B_2$ and B_2 such that $B_2 \cap K_{Y_3} = \varnothing$. That is, $e_4 \notin K_{Y_3}$. Notice that $e_4 \neq e_2$ since the cocircuit induced by e_2 and B_2 is X_1 and $e_2 \in K_{X_1}$ by definition. We have $B_1 \cap K_{Y_3} = \{e_1\}$ since otherwise, we would have $B_1 \cap K_{Y_3} = \varnothing$ contradicting $B_1 \in \mathcal{B}[\Omega]$ by Property 4. Notice that since $Y_3 \cap B_1 = \{e_1, e_4\}$, by Property 2, we have $e_4 \in C$.
Let $d = \arg\min\{\underline{\omega}_e \mid e \in X_1\}$ be a minimum element in X_1.
We show that d does not dominate e_3. If $e_2 = e_3$, then $X_2 = X_1$ and since $e_2 \in K_{X_1}$, d does not dominate e_3.
Otherwise, let X_3 be the cocircuit induced by e_3 and B_1. If $d \in X_3$, then since $e_3 \in K_{X_3}$, by Property 4, d can not dominate e_3. If $d \in X_1 \setminus X_3$, then by Lemma 4, $d \in X_2$. By hypothesis, $e_3 \in K_{X_2}$ and so, e_3 can not dominate e_3. Now, by definition of e_3, we have $\underline{\omega}_{e_3} \geq \underline{\omega}_{e_4}$. So, since e_3 is not dominated by d, e_4 is not dominated by d. It follows that no element in X_1 dominates e_4. Let X_4 be the cocircuit induced by e_4 and B_1. Since $\{e_2, e_4\} \subset C$, we have $e_2 \in X_4$. By Property 4, $e_4 \in K_{X_4}$ and so, no element of X_4 can dominate e_4. Hence, no element of $X_1 \cup X_4$ dominates e_4, and by Lemma 4, no element of $Y_3 \subseteq X_1 \cup X_4$ dominates e_4. That is, e_4 belongs to K_{Y_3}, which contradicts $B_2 \cap K_{Y_3} = \varnothing$. We can conclude that B_2 is not dominated.

□

Notice that if $e_1 = e_3$, then $B_3 = B_2$ and thus, B_2 is necessarily non-dominated. Moreover, if $e_2 = e_3$, the condition of property Lemma 5(b) is fullfiled and so, B_2 is non-dominated.

4 Core and partial solution

Finally, we show that the property of being possible/necessary for an element is intrinsically linked to the concept of cores. We first introduce some definitions.

Definition 5. *Let e be an element, let S be a nice partial solution and let X be a cocircuit such that $e \in K_X$.*

- *X is a* necessary certificate *for e if $K_X \setminus out(S) = \{e\}$.*
- *Let Y be a cocircuit, Y is a* bad cocircuit *for e and X if $e \notin K_Y$ and $K_Y \setminus out(S) \subseteq X$.*
- *X is a* possible certificate *for e if $in(S) \cap X = \varnothing$ and there is no bad cocircuit Y for e and X.*

As we will show in the following, an element e is necessary with respect to a nice partial solution S if and only if there exists a necessary certificate for e. Samewise, an element e is possible with respect to a nice partial solution S if and only if there exists a possible certificate for e. An example of necessary and possible certificates is depicted in Example 9.

Example 9. Consider the imprecise weighted matroid $M = (E, \mathcal{B})$ described in Example 2 and consider the partial solution S with $in(S) = \varnothing$ and $out(S) = \{e_1\}$.

- The cocircuit $X_1 = \{e_1, e_4, e_6\}$ is a necessary certificate for e_4 since $K_X = \{e_1, e_4\}$ and $K_X \setminus out(S) = \{e_4\}$.
- The cocircuit $X_2 = \{e_3, e_4, e_5\}$ is not a possible certificate for e_3 since X_1 is a bad cocircuit for e_3 and X_2.
- The cocircuit $X_3 = \{e_2, e_3\}$ is a possible certificate for e_3 since $e_3 \in K_{X_3}$, $X_3 \cap in(S) = \varnothing$ and there is no bad cocircuit for e_3 and X_3.

4.1 Necessary certificate

We show in this part that an element is necessary with respect to a partial solution if and only if there is a necessary certificate for it. We introduce Algorithm 2 that, given a nice partial solution S and an element e such that is no necessary certificate for e, construct a base associated to S that does not contain e. An example for Algorithm 2 is depicted in Figure 3.

Algorithm 2: not_necessary

Data:
- $M = (E, \mathcal{B}, \Omega)$: an imprecise weigthed matroid.
- S: a nice partial solution.
- e_1 : an element such that there is no necessary certificate for e.
- B_1: a base associated with S such that $e_1 \in B_1$.
- Y: a subset of forbidden elements such that $B_1 \cap Y = \varnothing$ and there is no cocircuit X with $K_X \setminus (out(S) \cup Y) = \{e_1\}$.

Result: a base associated to S that does not contain any element of $Y \cup \{e_1\}$.

$i \leftarrow 0$;
$B_1^i \leftarrow B_1$;
Loop
  Let X_1^i be the cocircuit induced by e_1 and B_1^i ;
  Let e_2^i be an element in $K_{X_1^i} \setminus (\{e_1\} \cup out(S) \cup Y)$;
  Let C^i be the circuit in $B_1^i \cup \{e_2^i\}$;
  Let $e_3^i = \arg\max\{\underline{\omega}_e \mid e \in C^i\}$;
  Let B_2^i be the base obtained by swapping e_1 and e_2^i in B_1^i ;
  Let B_3^i be the base obtained by swapping e_2^i and e_3^i in B_1^i ;
  if B_2^i *is not dominated* **then**
    return B_2^i ;
  $B_1^{i+1} = B_3^i$ // we repeat the process with the base B_3^i;
  $i \leftarrow i + 1$;

Lemma 6. *Algorithm 2 is correct, that is, the returned base is associated to S and does not contain any element of $Y \cup \{e_1\}$.*

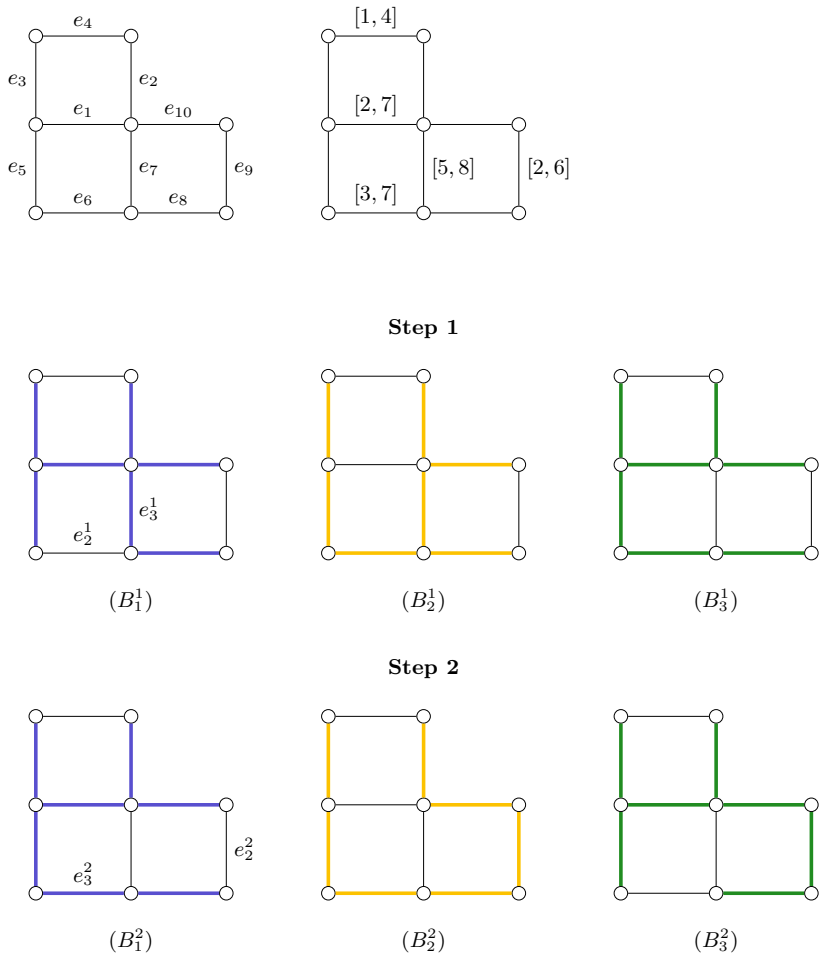

Fig. 3. Example of application of Algorithm 2 in minimum spanning trees. We have $out(S) = \{e_4\}$ and $in(S) = \{e_2, e_3, e_5, e_8, e_{10}\}$. Any edge $e \in in(S)$ has $\Omega(e) = [1, 1]$. There exits no necessary certificate for e_1 and we want to show that e_1 is not necessary with respects to S. **Step 1.** The algorithm starts with input tree B_1^1. We have $X_1^1 = \{e_1, e_4, e_6\}$ and $e_6 \in K_{X_1^1}$. So, we set $e_2^1 = e_6$. The maximum edge in the cycle $C^1 = \{e_1, e_5, e_6, e_7\}$ is the edge e_7, so we set $e_3^1 = e_7$. The bases B_2^1 and B_3^1 are obtained by swapping e_2^1 with, respectively e_1 and e_3^1 in B_1^1. The base B_2^1 is dominated because the edge $d = e_4$ dominates e_7 in the cocircuit $X_2^1 = \{e_1, e_4, e_7\}$. The base B_3^1 is not dominated by Lemma 5. We repeat the process with input tree $B_3^1 = B_1^2$. **Step 2.** We have $X_1^2 = \{e_1, e_4, e_9\}$ and $e_9 \in K_{X_1^2}$. So, we set $e_2^2 = e_9$. The maximum edge in the cycle $C^2 = \{e_1, e_5, e_6, e_8, e_9, e_{10}\}$ is the edge e_6. So, we set $e_3^2 = e_6$. The bases B_2^2 and B_3^2 are obtained by swapping e_2^2 with, respectively e_1 and e_3^2 in B_1^2. The edge e_6 belongs to the core $K_{X_2^2}$ of the cocircuit $X_2^2 = \{e_1, e_4, e_6\}$. Hence, by Lemma 5, B_2^2 is non-dominated. We obtain a non-dominated base associated to S that does not contain e_1 and thus, e_1 is not necessary with respect to S.

Proof. We suppose that the input is correct, that is, each parameter respects the conditions specified by Algorithm 2. For any step i, let d_i be the minimum element of X_1^i, that is, $d_i = \arg\min\{\underline{\omega}_e \mid \in X_1^i\}$. We show the following claims.

Claim 1: at each step i of the loop, we have $B_1^i \in \mathcal{B}_S[\Omega]$.
By construction, we have $B_1 = B_1^0$ and thus, $B_1^0 \in \mathcal{B}_S[\Omega]$. Now, suppose by induction hypothesis, that $B_1^i \in \mathcal{B}_S[\Omega]$. We show that either Algorithm 2 returns B_2^i or, we have $B_3^i = B_1^{i+1} \in \mathcal{B}_S[\Omega]$. Notice that by hypothesis, the element e_2^i exists, since otherwise we would have $K_{X_1^i} \setminus (out(S) \cup Y) = \{e_1\}$. If $e_1 = e_3^i$, then $B_3^i = B_2^i$ and thus, by Lemma 5(a) B_2^i is non-dominated so, B_2^i is returned. If $\underline{\omega}_{e_2^i} = \underline{\omega}_{e_3^i}$, then by taking $e_3^i = e_2^i$, we have $B_2^i \in \mathcal{B}_S[\Omega]$ by Lemma 5(b). So, Algorithm 2 returns B_2^i. If $e_3^i \notin in(S)$, then $B_1^{i+1} = B_3^i \in \mathcal{B}_S[\Omega]$ by Lemma 5(a) Hence suppose that $e_1 \neq e_3^i$, $\underline{\omega}_{e_2^i} < \underline{\omega}_{e_3^i}$ and $e_3^i \in in(S)$. Since $e_3^i \in in(S)$ and since S is a nice partial solution, we have $\underline{\omega}_{e_2^i} = \underline{\omega}_{e_3^i}$ which is a contradiction. Hence, we can assume that $e_3^i \notin in(S)$ and thus, by Lemma 5(a), $B_3^i \in \mathcal{B}_S[\Omega]$.

Claim 2: at each step i of the loop, we have $B_1^i \cap Y = \varnothing$.
By construction, we have $B_1 = B_1^0$ and thus, $B_1^0 \cap Y = \varnothing$. Now, suppose by induction hypothesis that $B_1^i \cap Y = \varnothing$. By construction, we have $B_1^{i+1} \setminus B_1^i = \{e_2^i\}$. Since $e_2^i \notin Y$, it follows that $B_1^{i+1} \cap Y = \varnothing$.

Claim 3: at each step i of the loop, we have $\underline{\omega}_{d_i} \leq \underline{\omega}_{d_{i+1}}$.
Let X_3^i and Y^i be the cocircuits induced by e_3^i and B_1^i and B_2^i, respectively. Since B_2^i is not returned, there is an element $f \in Y^i$ that dominates e_3^i. By Lemma 4, $f \in X_1^i \cup X_3^i$. The element f can not belong to X_3^i since otherwise, we would have $K_{X_3^i} \cap B_1^i = \varnothing$ contradicting B_1^i being non-dominated by Property 4. Thus, f belongs to X_1^i and so, no element of $X_1^i \cup X_3^i$ dominates d^i. By Lemma 4, $X_1^{i+1} \subseteq X_1^i \cup X_3^i$, thus, no element of X_1^{i+1} dominates d^i. It follows that $\underline{\omega}_{d_i} \leq \underline{\omega}_{d_{i+1}}$.

Claim 4: the algorithm stops, that is, Algorithm 2 returns B_2^i at some step i.
Toward a contradiction, suppose the loop is infinite. Rougly speaking, it means that there is an element x that is swapped in during a step j and the same element x is swapped out during another step k. That is, there are two steps j and k, with $j < k$ such that $e_3^k = e_2^j$. We show that in that case, Algorithm 2 returns B_2^k during the step k.
Now suppose that there are two steps j and k of the loop with $j < k$ such that $e_3^k = e_2^j$. Since e_2^j is considered at step j, we have $e_2^j \in K_{X_1^j}$, that is, e_2^j is not dominated by d^j nor d^k. The element e_3^k can not be dominated by an element of $K_{X_3^k}$ since otherwise we would have $K_{X_3^k} \cap B_1^k = \varnothing$ contradicting B_1^k being non-dominated by Property 4. Hence, e_3^k is not dominated by an element of $X_1^k \cup X_3^k$ and by Lemma 4, we have $e_3^k \in K_{Y^k}$. It follows that, by Lemma 5(b) B_2^k is non-dominated and thus, B_2^k is returned by the Algorithm 2 at step k.

Hence, by combining the previous claims, we can conclude that Algorithm 2 returns a base, which finishes the proof. □

Hence, we can deduce the following corollary.

Corollary 2. *Let S be a nice partial solution. Let $e \notin in(S) \cup out(S)$ be an element, e is necessary with respect to S if and only if there is a necessary certificate for e.*

Proof. Let X be a cocircuit such that $K_X \setminus out(S) = \{e\}$. Then, for any base $B \in \mathcal{B}_S[\Omega]$, we have $e \in B$ since otherwise it contradicts Property 4. Thus, e is necessary with respect to S. We now show the reciprocity. Let $e \notin in(S) \cup out(S)$ be an element and let assume that there is no cocircuit X such that $K_X \setminus out(S) = \{e\}$. Let B be a base of $\mathcal{B}_S[\Omega]$. If $e \notin B$ then e is not necessary with respect to S. Otherwise, by Lemma 6, Algorithm 2 with input M, S, B, e and $Y = \varnothing$ returns a base associated to S that does not contain e. So, we can conclude that e is not necessary. □

4.2 Possible certificate

In this part, we show that an element is possible with respect to a nice partial solution if and only if there exists a possible certificate for it. We introduce Algorithm 3 that, given a partial solution S, an element e with a possible certificate for it, constructs a base associated to S containing e. An example for Algorithm 3 is depicted in Figure 4.

Algorithm 3: is_possible

Data:
- $M = (E, \mathcal{B}, \Omega)$: an imprecise weigthed matroid.
- S: a nice partial solution.
- e : an element of M.
- X: a possible certificate for e.
- B: a base associated to S.

Result: a base associated to S that contains e.

```
1 i ← 0 ;
2 B^i ← B;
3 Loop
4 |   if e ∈ B^i then
5 |   |   return B^i ;
6 |   Let x^i be an element that is X-blocking for e in B^i;
7 |   Y^i ← X \ (B^i ∪ {e});
8 |   B^{i+1} = not_necessary(M, S, x^i, B^i, Y^i) // we repeat the process with a
  |     base that does not contain x^i ;
9 |   i ← i + 1 ;
```

Lemma 7. *Algorithm 3 is correct, that is, the returned base is associated to S and contains e.*

Proof. We suppose that the input given to the algorithm is correct, that is, each parameter respects the conditions specified by Algorithm 3.

First, we show that at each step i, the call of the function *not_necessary* at line 8 is correct. That is, all parameters given as input respect the conditions specified by Algorithm 2. By definition, M and S respect the specified conditions. By construction of B^i, we have $x^i \in B^i$. Further, there is no cocircuit X' such that $K_{X'} \setminus out(S) = \{x^i\}$, since otherwise we would have $K_{X'} \setminus out(S) \subset X$ which is a contradiction. By construction of Y^i, we have $Y^i \cap B_2^i = \varnothing$. Finally, suppose that there is a cocircuit X' such that $K_{X'} \setminus (out(S) \cup Y^i) = \{x^i\}$. Since, $Y^i \subset X$ and $x^i \in X$, we have $K_{X'} \setminus out(S) \subset X$, which is a contradiction. Hence, we can conclude that all parameters respect the specified conditions.

Further, we show that the algorithm stops. Let $W = X \setminus \{e\}$. By Lemma 6, at each step i, we have $x^i \notin B^{i+1}$ and $Y^i \cap B^{i+1} = \varnothing$. By construction of Y^i, we have $W \cap B^{i+1} = W \cap B^i \setminus \{x^i\}$ (no element of W can be added to B^i to construct B^{i+1}). It follows that $|W \cap B^i| > |W \cap B^{i+1}|$, and so there is at most $|W|$ loop steps. Notice that after $|W|$ steps, we necessarily have $B^i \cap W = \varnothing$.

Finally, by Algorithm 2, at each step i, B^i is not dominated. By Property 4, we have $K_X \cap B^i \neq \varnothing$. Hence, since after $|W|$ steps B^i and W are disjoint and since $K_X \setminus W = \{e\}$, it implies that there is a step $i \leq |W|$ such that $e \in B^i$. We can conclude that the algorithms returns a base as thought. □

From Lemma 7, we can infer the following corollary.

Corollary 3. *Let S be a nice partial solution. Let $e \notin in(S) \cup out(S)$ be an element, e is possible with respect to S if and only if there is a possible certificate for e.*

Proof. Let e be an element and let X be a possible certificate for e. Let B be a base associated to S. If $e \in B$, then e is possible with respect to S. Otherwise, by Lemma 7, Algorithm 3 with input M, S, X, e an B returns a base associated to S that contains e. So, e is possible with respect to S. We now show the reciprocity. Let $e \notin in(S)$ be an element and suppose that there is no possible certificate for e. Toward a contradiction, suppose e is possible with respect to S. Let B be a base associated to S that contains e. Let X be the cocircuit induced by e and B. Notice that $in(S) \cap X = \varnothing$ as $e \notin in(S)$. Since X is not a possible certificate, there is necessarily another cocircuit Y that is bad for X and e. But then, we have $K_Y \cap B = \varnothing$, which is a contradiction by Property 4. Thus, we can conclude that e is not possible with respect to S. □

5 Enumerating algorithm

Having stated our formal results, we are now ready to provide our enumerating algorithms relying on them. In this section, we use Corollaries 2 and 3 to develop two algorithms that determine if an element e is possible/necessary with respect to a given nice partial solution. Informally, the principle of the algorithms is to observe/check if e closes a circuit in some specific independent sets.

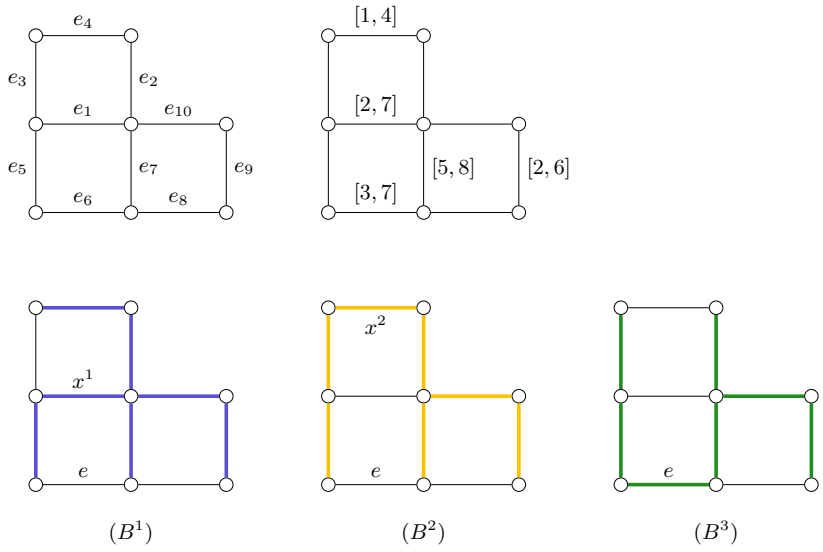

Fig. 4. Example of application of Algorithm 3 in minimum spanning trees. We have $out(S) = \{e_8\}$ and $in(S) = \{e_2, e_5\}$. Any edge $e \in in(S) \cup out(S)$ has $\Omega(e) = [1, 5]$. The cocircuit $X = \{e_1, e_4, e_6\}$ is a possible certificate for the edge $e = e_6$. **Step 1.** The algorithm starts with input tree B^1. The edge $x^1 = e_1$ is X-blocking for e in B^1. We set $Y^i = \varnothing$ and we call $not\_necessary(M, S, x^1, B^1, Y^1)$ to obtain a tree B^2 that does not contain x^1. **Step 2.** We repeat the process with the tree B^2. The edge $x^2 = e_4$ is X-blocking for e. We set $Y^2 = \{e_1\}$ and we call $not\_necessary(M, S, x^2, B^2, Y^2)$ to obtain a tree B^3 that does not contain x^2 nor an edge in Y^2. **Step 3.** The tree B^3 contains e and so, B^3 is returned at this step. Thus, e is possible with respect to S.

5.1 Possible element of nice partial solutions

We now show how to determine if a first unassignated element is possible with respect to a nice partial solution S. To see if a first unassignated element e is possible with respect to a nice partial solution, we simply need to check if the addition of e inside a specific subset of elements closes a circuit, as shown by Algorithm 4. The difficulty here is to show that if the algorithm returns true, then there exists a cocircuit that is a possible certificate for e. To do this, we introduce Algorithm 5 which produces such possible certificate. An example for Algorithm 5 is given by Figure 5.

Lemma 8. *Algorithm 5 is correct.*

Proof. First, note that at each step i, the subset of elements E^i exists by Lemma 3. Further, we show that at each step i, we have $e \in K_{X^i}$. Suppose not, and let x be element in $K_{X^i} \setminus out(S)$. Note that by hypothesis $x \notin in(S)$, since $in(S) \subseteq E^i$. Since $e \notin K_{X^i}$ and $x \in K_{X^i}$, it implies that $\underline{\omega}_x < \underline{\omega}_e$ contradicting that E' contains every element that dominates e. Thus, $e \in K_{X^i}$ at each step i.

Algorithm 4: check_is_possible

Data: An imprecise weighted matroid $M = (E, \mathcal{B}, \Omega)$, a nice partial solution S and a first unassignated element e of S.
Result: `true` if e is possible with respect to S, `false` otherwise.

```
Let E' = {e' ∈ E | e' dominates e} ∪ in(S) ∪ {e};
if there is a circuit in C(E') containing e then
    return false;
return true;
```

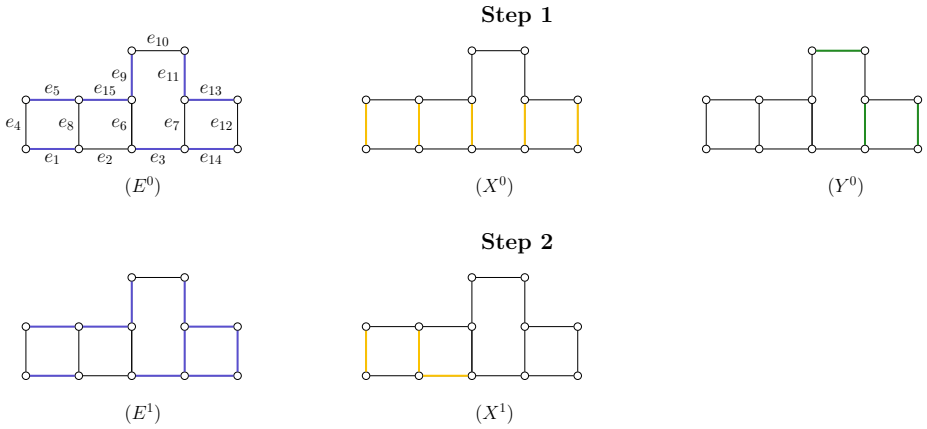

Fig. 5. Example of application of Algorithm 5 in minimum spanning trees. We have $out(S) = \{e_{10}\}$ and $E^0 = in(S) = \{e_1, e_3, e_5, e_9, e_{11}, e_{13}, e_{14}, e_{15}\}$. We have $W(e_4) = [4, 6]$, $W(e_{10}) = [1, 3]$, $W(e) = [2, 5]$ for any edge in $\{e_6, e_7, e_8, e_{12}\}$ and $W(e) = [1, 3]$ for any other edge. We want to construct a possible certificate for the edge e_4. **Step 1.** The algorithm starts with $X^0 = \{e_4, e_6, e_7, e_7, e_{12}\}$. There exists a bad cocircuit $Y^0 = \{e_7, e_{10}, e_{12}\}$ for e_4 and X^0: indeed we have $Y^0 \setminus out(S) = \{e_7, e_{12}\} \subset X^0$. Hence, we have $W^0 = \varnothing$ and thus, we set $E^1 = E^0 \cup \{e_7, e_{12}\}$. We also obtain the cocircuit $X^1 = \{e_2, e_4, e_8\}$ that contains $W^0 \cup \{e_4\}$ and avoid E^1. **Step 2.** There is no bad cocircuit for e and X^1 and thus, X^1 is returned at this of the algorithm. Hence, X^1 is a possible certificate for e.

Now we show that the algorithm stops at some point. We first show that for each step i, if X^i is not a possible certificate for e, then $|E^i| > |E^{i-1}|$. Notice that we can not have $|E^i| < |E^{i-1}|$ since $E^{i-1} \subseteq E^i$ by construction. Suppose there is a step i such that $|E^i| = |E^{i-1}|$. Then, in that case we have $Y^{i-1} \setminus W^{i-1} = \varnothing$. Thus, we obtain $Y^{i-1} \subseteq X^i$, by definition of a cocircuit we have $Y^{i-1} = X^i$ since otherwise it contradicts the minimality of X^i. But then, since Y^{i-1} is a bad cocircuit for X^{i-1} and e, it implies that $e \notin K_{Y^i - 1}$ which is a contradiction. Hence, we have $|E^i| > |E^{i-1}|$. Since the number of element in M is finite, it also implies that the algorithm stops at some point.

□

Algorithm 5: construct possible_certificate_cocircuit

Data:
- $M = (E, \mathcal{B}, \Omega)$: an imprecise weighted matroid.
- S: a nice partial solution.
- e: element not in $in(S) \cup out(S)$.
- X: such that $e \in K_X$ and $in(S) \cap X = \varnothing$.
- E': a set of elements $X \cap E' = \varnothing$.

Result: If $in(S) \subseteq E'$ and E' contains every elements not in $out(S)$ that dominates e, then the algorithm returns a possible certificate for e.

1 $i \leftarrow 0$;
2 $E^0 \leftarrow E'$;
3 $X^0 \leftarrow X$;
4 **while** *There is a bad cocircuit* Y^i *for* e *and* X^i **do**
5 $\quad W^i \leftarrow \{e' \in K_{Y^i} \mid \exists C \in \mathcal{C}(E^i \cup \{e, e'\})$ `s.t.` $\{e, e'\} \subseteq C\}$;
6 $\quad E^{i+1} \leftarrow E^i \cup Y^i \setminus W^i$;
7 $\quad$ Let X^{i+1} be the cocircuit containing $W^i \cup \{e\}$ and avoiding E^{i+1};
8 $\quad i \leftarrow i + 1$;
9 **return** X^i;

Lemma 9. *Algorithm 4 is correct. Hence, given a nice partial solution S, we can determine if a first unassignated element is possible with respect to S in $\mathcal{O}(|E| + p_M)$ or in $\mathcal{O}(|E| \cdot i_M)$, depending on the used oracle.*

Proof. We show that Algorithm 3 returns `true` if and only if e is possible with respect to S. First, suppose that the algorithm returns `true`, that is, there is no circuit C in E' such that C contains e. By Lemma 3, there is a cocircuit X such that $e \in X$ and $X \cap E' = \varnothing$. By Lemma 8, Algorithm 5 with input M, S, e, X and E' returns a possible certificate for e. Hence, by Corollary 3, e is possible with respects to S.

Now suppose that e is possible with respect to S. By Corollary 3, there is a cocircuit X that is a possible certificate for e. That is, no element of X dominates e. By construction of E', no element of X belongs to $E' \setminus \{e\}$. Thus, by Property 2 for any circuit C containing e, we have $C \not\subseteq E'$. Hence, E' does not contain a circuit containing e and thus, the algorithm returns `true`.

Concerning the running complexity of the algorithm: E' is constructed in $\mathcal{O}(|E|)$, then there is one call of the port oracle in line 2. Hence, we obtain a time complexity $\mathcal{O}(|E| + p_M)$. If we do not use the port, then by Remark 1, we have time complexity $\mathcal{O}(|E| \cdot i_M)$, using the independence oracle. □

Notice that, we can also use Algorithm 4 to determinate if an element is possible with respect to the empty partial solution $S_\varnothing$. Since there is no need to sort the elements by increasing order of weight if the partial solution is empty, Algorithm 3 has a better time complexity than the one developed by Kasperski and Zieliński [12] to determine if an element is possible (*i.e.* if we run Algorithm 3 with $S := S_\varnothing$).

5.2 Necessary element of nice partial solutions

We now show how to determinate if an element is necessary with respect to a nice partial solution S. Notice that, contrary to Algorithm 4 that only works for first unassignated elements (or if $S = S_\varnothing$), Algorithm 6 works for any element $e \notin in(S) \cup out(S)$.

Algorithm 6: check_is_necessary

Data: An imprecise weighted matroid $(E, \mathcal{B}, \Omega)$, a nice partial solution S and an element e.
Result: `true` if e is necessary with respect to S, `false` otherwise.

```
1  Let E' such that E' = {e' ∈ E \ {e} | e does not dominate e'};
2  if there is no circuit containing e in E' ∪ {e} then
3  |  return true;
4  forall e_i ∈ out(S) such that e_i does not dominate e do
5  |  Let E^i such that
   |    E^i = {e' ∈ E | e_i does not dominate e'} \ (out(S) ∪ {e});
6  |  if there is no circuit containing e in E^i ∪ {e}, and
7  |  there is no circuit containing e_i in E^i ∪ {e_i}, and
8  |  there is a circuit containing e and e_i in E^i ∪ {e, e_i}, then
9  |  |  return true;
10 return false;
```

Lemma 10. *Algorithm 6 is correct. Hence, we can determine if an element is necessary with respect to a nice partial solution S in $\mathcal{O}((p_M + |E|) \cdot (|out(S)| + 1))$ or in $\mathcal{O}(i_M \cdot |E| \cdot (|out(S)| + 1))$, depending on the used oracle.*

Proof. We show that Algorithm 6 returns `true` if and only if e is necessary with respect to S. First, suppose that the algorithm returns `true`. Two cases can occur.

1. Algorithm 6 returns `true` at line 3. That is, there is no circuit containing e in E'. By Lemma 3, there is a cocircuit X such that $e \in X$ and $E' \cap X = \varnothing$. Any element $e' \neq e$ in X is dominated by e since otherwise e' would belong to E'. Hence, we have $K_X = \{e\}$ and so, X is a necessary certificate for e. By Corollary 2, e is necessary with respect to S.
2. Algorithm 6 returns `true` at line 8. That is, there is an element $e_i \in out(S)$ that does not dominate e and such that (a) there is no circuit containing e in E^i, and (b) there is a circuit in $C \subseteq E^i$ containing e and e_i. By Lemma 3, there is a cocircuit X such that $\{e, e_i\} \subseteq X$ and $E^i \cap X = \varnothing$.
   Further, note that no element $e' \in X \setminus (out(S) \cup \{e\})$ belongs to K_X since otherwise e' would not be dominated by e_i and would belong to E_i. Suppose that $e \notin K_X$, then we have $K_X \subseteq out(S) = \varnothing$. So, any base B such that $in(S) \subseteq B$ and $out(S) \cap B = \varnothing$ does not belong to $\mathcal{B}[\Omega]$ since it is not

possible to respect Property 4. Thus, no non-dominated base is associated to S, contradicting that S is a partial solution. Hence, $e \in K_X$ and we have $K_X \setminus out(S) = \{e\}$, that is, X is a necessary certificate for e. By Corollary 2, e is necessary with respect to S.

Now suppose that e is necessary with respect to S. By Corollary 2, there is a cocircuit X that is necessary certificate for e. If $K_X = \{e\}$, then no element of X belongs to the set E' defined at line 1. By Property 2, any circuit containing e contains another element in X. Thus, $E' \cup \{e\}$ does not contain a circuit containing e. So Algorithm 6 returns `true` at line 3. Now suppose that K_X contains more than one element and let $e_X \neq e$ be an element of X that dominates every element in $X \setminus K_X$. We have $e_X \in out(S)$. Let E^i be the set defined at line 5 when $e_i = e_X$. By definition of E^i, we have $X \cap E^i = \varnothing$. Thus, by Property 2, $E^i \cup \{e\}$ and $E^i \cup \{e_i\}$ do not contain a circuit containing e and e_i, respectively. It remains to show that there is a circuit in $E^i \cup \{e, e_i\}$ containing e and e_i. Toward a contradiction, suppose not. Let $B \in \mathcal{B}_S[\Omega]$ be a non-dominated base associated to S. Since $e_i \in out(S)$, e_i does not belong to B. Let C be the circuit created by the addition of e_i in B. By Property 2, there is an element e' in $X \cap C \cap B$. If $e' \neq e$, then $e' \notin K_X$ and thus, e_i dominates e'. So, the base obtained by swapping e' and e_i in B dominates B which is a contradiction. Now suppose that $e' = e$. There is an element e'' in C such that $e'' \notin E_i$. By definition of E_i, e'' is dominated by e_i and so, the base obtained by swapping e'' and e_i in B dominates B which is a contradiction.

Finally, concerning the time complexity of the algorithm: E' can be constructed in $\mathcal{O}(|E|)$, then a call to the port oracle is made in line 2. So, lines 1 to 3 involve a time complexity of $\mathcal{O}(|E| + p_M)$. For each element $e_i \in out(S)$, E^i can be constructed in $\mathcal{O}(|E|)$, then three calls to the port oracle are made in the lines 6 to 8. So, we obtain a time complexity $\mathcal{O}((p_M + |E|) \cdot |out(S)|)$ for the loop at lines 4 to 9. Thus, the time complexity for the entire algorithm is $\mathcal{O}((p_M + |E|) \cdot |out(S)| + 1)$. If we do not use the port oracle, then by Remark 1, we obtain a time complexity $\mathcal{O}((i_M \cdot |E|) \cdot |out(S)| + 1)$ using the independence oracle. □

Notice that, once again, since there is no need to sort the elements by increasing order of weight if the partial solution is empty, Algorithm 6 has a better time complexity than the one developed by Kasperski and Zielińsky [12] to determine if an element is necessary. Indeed, if we run Algorithm 6 with $S := S_\varnothing$, then the time complexity is $\mathcal{O}(p_M + |E|)$ or $\mathcal{O}(i_M \cdot |E|)$.

5.3 The enumerating algorithm

Now that we developed two polynomial-time algorithms to determine if a first unassignated element is possible/necessary with respect to some partial solution, we can enumerate every base of $\mathcal{B}[\Omega]$ with an exhaustive search.

Corollary 4 (Lemma 9 and Lemma 10). *Algorithm 7 is correct. Hence, $\mathcal{B}[\Omega]$ can be enumerated in $\mathcal{O}(b \cdot (|E|^2 \cdot p_M + |E|^3))$ (or in $\mathcal{O}(b \cdot (|E|^3 \cdot i_M))$ if the port oracle is not allowed), where $b = |\mathcal{B}[\Omega]|$.*

Proof. We show that the time complexity is correct. First, the list L is created in $\log|E|$. Then, at each call of the enumeration, the two functions *is_possible* and *is_necessary* are called on each element in $E \setminus (out(S) \cup in(S))$. So, each call, without taking in account the recursive call, has a complexity of $\mathcal{O}((|E| + p_M) \cdot (|out(S)| + 1)) = \mathcal{O}(|E| \cdot p_M + |E|^2)$. We need $|E|$ reccursive calls to display one non-dominated base, that is, the complexity for one enumeration is $\mathcal{O}((|E|^2 \cdot p_M + |E|^3))$. Finally, since there are b bases to enumerate, we obtain a time complexity of $\mathcal{O}(b \cdot (|E|^2 \cdot p_M + |E|^3) + \log|E|) = \mathcal{O}(b \cdot (|E|^2 \cdot p_M + |E|^3))$. If the port oracle is not allowed, by Remark 1, we have a time complexity $\mathcal{O}(b \cdot (|E|^3 \cdot i_M))$ by using the independence oracle. □

Algorithm 7: Enumeration Algorithm

```
1  Function Enumeration(M):
       Data: An imprecise weighted matroid M = (E, B, Ω)
2      S ← S_∅
3      L ← sort E by increasing order of ω_e
4      EnumerationRec(M,S,L)
5
6  Function EnumerationRec(M, S, L):
       Data: An imprecise weighted matroid M = (E, B, Ω), a nice partial
             solution S and L the sorted list of elements in E \ in(S) ∪ out(S).
7      if L = ∅ then
8          Display(in(S))
9      else
10         e ← first element of L
11         L ← L \ {e}
12         if not check_is_possible(M, e, S) then
13             S' ← S
14             out(S') ← out(S') ∪ {e}
15             EnumerationRec(M, S', L)
16         else
17             S' ← S
18             in(S') ← in(S') ∪ {e}
19             EnumerationRec(M, S', L)
20             if not check_is_necessary(M, e, S) then
21                 out(S) ← out(S) ∪ {e}
22                 EnumerationRec(M, S', L)
```

6 Numerical Experiments

In this section, we present some tests on random generated instances. For this section, we were interested in the problem of finding all non-dominated spanning trees of a given graph, which is a specific yet well-studied instance of Matroids

(where E are the set of edges of the graphs, and the bases all the spanning trees of the graph, that are of constant size). The source code and the instances are available at `https://gitlab.utc.fr/davottom/enum-imst`. We compare Algorithm 7 with the two following methods.

- **Outer approximation.** This method first compute a subgraph G' constituted by the possible and necessary edges in the initial graph. Then, it enumerates every spanning trees of G' that contains all necessary edges.
- **Reduce.** This method uses same algorithm than the outer approximation plus check for each spanning tree T of G' if T is non-dominated. To check if a tree T is non-dominated, we use the same idea as the one described in Theorem 1: we compute a minimum spanning tree in the realization R where $R(e) = \underline{\omega}_e - \epsilon$ if $e \in E(T)$ and, $R(e) = \overline{\omega}_e$, otherwise.

In the following, we refer to Algorithm 7 as the exact method.

6.1 Instances

We generated imprecise weighted graphs with 12 vertices by varying the density of the graph and the weight function. We chose to generate the instances according to three graph densities and three scenarios for the weight function. The three possible densities *sparse*, *middle*, *dense* for which the graph contains $18, 30$ and 42 edges, respectively. The graph is generated using the random generator of the library boost in C++. If the graph is not connected, we add a random edge between two connected components until the graph is connected. For the generation of weight functions, given a scenario i for each edge e, we pick two random numbers $\ell \in [1 - 10]$ $s \in [a_i, b_i]$, where a_i and b_i depend on the selected scenario. Then, we set $\Omega(e) = [\ell, \ell + s]$. For scenario 1, we have $a_i = 1$ and $b_i = 10$, for scenario 2, we have $a_i = 7$ and $b_i = 9$ and, for scenario 3, we have $a_i = 2$ and $b_i = 3$. Note that scenario 1 generates intervals with quite varying sizes, while scenario 2 generates intervals that will very often overlap. For each scenario and each density, we generate 10 instances.

6.2 Results

The tests were run on a personal computer with 32Go of RAM and with an Intel Core 7 processor 5.20Ghz. The results are depicted in Tables 1 and 2. The detailled statistics for scenario 2 are depicted in Table 3. We can first observe that in scenarios 1 and 3, the exact method is the fastest method or, has an execution time almost equivalent to the other two methods for the sparse densities. For scenario 2, the results are more contrasted. In dense graphs, the exact method is the fastest for 4 instances, the second fastest for 2 instances and the slowest for 4 instances. In graphs with a middle density, the exact method is the fastest for 3 instances, the second fastest for 2 instance and the slowest for 5 instances. In sparse graphs, the running times are roughly the same, except for 2 instances where the exact method is significantly slower. Not surprisingly, the running

time of the exact method has better running time compare to the running time of the outer approximation if the difference between the number of enumerated spanning trees is high, as shown by Figure 6. Regarding the statistics on the number of trees enumerated, the denser the graph, the bigger the cardinality of the enumerated sets for both methods. Samewise, the larger the intervals (*i.e.* in scenario 1), the bigger the cardinality of the enumerated sets. We can also observe than when the graph is not dense, the outer approximation seems reasonably close to the exact method.

Table 1. Time statistics. A set contains every graphs generated with the same density and scenario. For each set and each method, average, minimum and maximum times are depicted.

Set		Exact			Approx			Reduce		
density	scenario	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
dense	1	1 s	71 ms	7 s	24 s	357 ms	1 m	33 s	510 ms	2 m
middle	1	120 ms	10 ms	610 ms	912 ms	25 ms	3 s	1 s	32 ms	4 s
sparse	1	1 ms	<1ms	5 ms	1 ms	<1ms	4 ms	1 ms	<1ms	5 ms
dense	2	7 m	52 s	17 m	6 m	4 m	7 m	8 m	6 m	9 m
middle	2	5 s	2 s	11 s	4 s	1 s	6 s	5 s	2 s	8 s
sparse	2	5 ms	1 ms	9 ms	3 ms	1 ms	6 ms	4 ms	2 ms	7 ms
dense	3	76 ms	1 ms	184 ms	323 ms	6 ms	828 ms	417 ms	6 ms	1 s
middle	3	2 ms	<1ms	9 ms	16 ms	<1ms	117 ms	19 ms	<1ms	135 ms
sparse	3	<1ms	<1ms	<1ms	<1ms	<1ms	<1ms	<1ms	<1ms	1 ms

Table 2. Result statistics on the number of enumerated tree. A set contains every graph generated with the same density and scenario. Exact and Approx: number of enumerated trees for the corresponding method. The Diff column is the difference of cardinality between the exact method and the outer approximation.

Set		Exact			Approx			Diff		
dens.	scen.	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
dense	1	904,332	17,909	1.7M	16M	229,596	70M	16M	197,940	69M
middle	1	85,267	3,648	213,070	676,426	15,921	2.6M	633,793	8,903	2.6M
sparse	1	1,178	78	2,148	1,009	117	3,641	420	0	1,928
dense	2	207M	13M	249M	242M	196M	280M	139M	31M	250M
middle	2	3.6M	944,655	4M	3.2M	1.2M	4,9M	1.5M	180,968	3.3M
sparse	2	4,580	696	3,868	2,840	1,435	4,150	550	0	1,567
dense	3	35,428	400	43,278	218,853	3,581	559,927	201,139	2,266	517,639
middle	3	1,930	34	4,384	11,927	54	84,830	10,962	20	80,446
sparse	3	247	10	280	292	13	794	169	0	514

Table 3. Statistic details for sets generated with scenario 2.

Scenario 2 - Dense							
Instance	Exact			Approx			Reduce
	Time	#Trees	Rank	Time	#Trees	Diff	Time
g_1	10m 14s	144M	3	5m 50s	228M	84M	7m 38s
g_2	17m 25s	249M	3	6m 54s	280M	31M	9m 10s
g_3	1m 31s	22M	1	6m 59s	272M	250M	9m 06s
g_4	6m 44s	96M	2	6m 43s	242M	146M	8m 35s
g_5	6m 26s	82M	2	6m 15s	242M	160M	8m 13s
g_6	0m 52s	12M	1	4m 53s	196M	184M	6m 23s
g_7	4m 26s	64M	1	5m 58s	228M	164M	7m 50s
g_8	5m 24s	78M	1	7m 18s	275M	196M	9m 33s
g_9	9m 00s	122M	3	5m 58s	228M	106M	7m 51s
g_10	11m 40s	164M	3	6m 6s	228M	64M	7m 55s

Scenario 2 - Middle							
Instance	Exact			Approx			Reduce
	Time	#Trees	Rank	Time	#Trees	Diff	Time
g_1	7s 751ms	2.5M	3	5s 620ms	4M	1.5M	7s 333ms
g_2	4s 287ms	1.5M	2	3s 681ms	2.7M	1.2M	4s 722ms
g_3	4s 550ms	1.5M	1	5s 715ms	4.1M	2.6M	7s 397ms
g_4	4s 329ms	1.5M	2	3s 980ms	3M	1.5M	5s 107ms
g_5	5s 679ms	2M	3	3s 452ms	2.8M	0.8M	4s 514ms
g_6	11s 762ms	4M	3	6s 595ms	4.9	0.9M	8s 605ms
g_7	2s 794ms	0.9M	1	4s 176ms	3M	2.1M	5s 382ms
g_8	4s 296ms	1.5M	1	6s 633ms	4.9M	3.4M	8s 426ms
g_9	3s 768ms	1.2M	3	2s 830ms	2M	0.8M	3s 633ms
g_10	2s 970ms	1M	3	1s 567ms	1.2M	0.2M	2s 066ms

Scenario 2 - Sparse							
Instance	Exact			Approx			Reduce
	Time	#Trees	Rank	Time	#Trees	Diff	Time
g_1	2ms	1,503	2	1ms	1,503	0	2ms
g_2	2ms	1,033	1	2ms	1,711	678	2ms
g_3	1ms	696	1	2ms	1,435	739	3ms
g_4	9ms	3,829	3	4ms	3,829	0	5ms
g_5	4ms	1,887	3	2ms	1,887	0	2ms
g_6	5ms	2,868	1	6ms	3,840	972	7ms
g_7	4ms	2,262	1	4ms	3,829	1,567	5ms
g_8	5ms	2,946	2	4ms	3,829	883	5ms
g_9	7ms	3,868	2	5ms	4,150	282	7ms
g_10	8ms	2,010	3	2ms	2,389	379	3ms

7 Conclusions

In this paper, we have discussed how sets of possibly optimal matroids $\mathcal{B}[\Omega]$ under interval-valued costs can be enumerated efficient, as given a partial solution, we have provided polynomial-time algorithms to check whether different

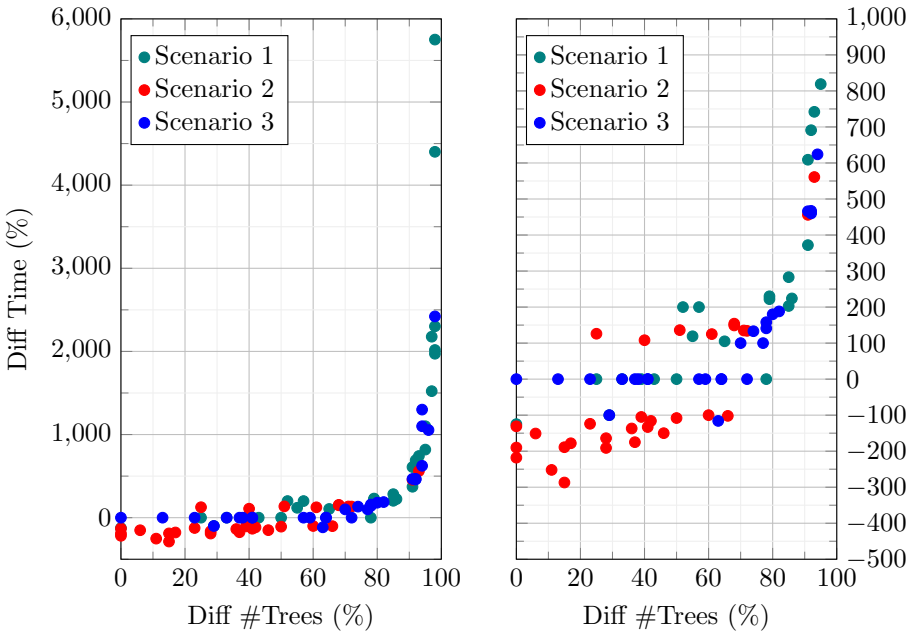

Fig. 6. Relation between the difference of number of enumerated trees and the difference of running time between the exact method and the approximation. The value of differences are displayed in percentage. A negative value for the time indicates that the approximation has a faster running time than the exact method. When the exact method has a better running time, the displayed value is equal to the running time of the approximation over the running time of the exact method. When the approximation has a better running time, the displayed value is equal to minus the running time of the exact method over the running time of the approximation. **Left:** contains all values. **Right:** contains values with a difference running time between -300% and 1000%.

elements are possible or necessary. What the paper also tells us is that this problem is in the case of Matroid very different from the simple problem of determining whether an element is in one or all of the solution of $\mathcal{B}[\Omega]$.

A natural further direction would be to consider more general uncertainty models, such as possibility distributions [8], belief functions [18] or credal sets [17,6]. However, the results in [6, Section 4] suggests that in optimisation problems where one mainly search to optimize sums of weights, either by minimizing (in the case of costs) or by maximizing (in the case of utilities) it, shifting to a credal setting does not modify significantly the problem, as lower/upper expectations remain linear, at least when every probability set is specified separately.

Another problem that we would like to revisit using the adopted point of view in this paper is the one of querying new information. Such problems have already been considered in the past for matroids, where one had to find an optimal (in

the sense of minimal total cost) set of queries that would lead to a unique optimal solution [14]. While such a view is interesting provided one is sure that all such queries can be performed, we would like to consider the problem in terms of incremental queries under uncertainty [1], where the budget is unknown and where the task is to choose the query that will result in the greatest uncertainty reduction (e.g., in terms of sets of possibly optimal solutions). Similar approaches have been sucessfully devised for matroids in the case where the (non-additive) objective function is unknown and has to be learned [3], letting us hope that we can develop the same kind of approaches for uncertain weights.

Acknowledgments

During this research, Tom Davot post-doctoral position was supported by the ANR project Preserve (ANR-18-CE23-0008).

References

1. Nadia Ben Abdallah and Sébastien Destercke. Optimal expert elicitation to reduce interval uncertainty. In *Uncertainty in Artificial Intelligence (UAI 2015)*, pages 12–22, 2015.
2. Ionut D. Aron and Pascal Van Hentenryck. On the complexity of the robust spanning tree problem with interval data. *Operation Research Letter*, 32(1):36–40, 2004.
3. Nawal Benabbou, Cassandre Leroy, Thibaut Lust, and Patrice Perny. Interactive optimization of submodular functions under matroid constraints. In *Algorithmic Decision Theory: 7th International Conference, ADT 2021, Toulouse, France, November 3–5, 2021, Proceedings 7*, pages 307–322. Springer, 2021.
4. Collette R. Coullard and Lisa Hellerstein. Independence and port oracles for matroids, with an application to computational learning theory. *Comb.*, 16(2):189–208, 1996.
5. Tom Davot, Sébastien Destercke, and David Savourey. On the enumeration of non-dominated spanning trees with imprecise weights. In *European Conference on Symbolic and Quantitative Approaches with Uncertainty*, pages 348–358. Springer, 2023.
6. Sébastien Destercke and Romain Guillaume. Necessary and possibly optimal items in selecting problems. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems: 19th International Conference, IPMU 2022, Milan, Italy, July 11–15, 2022, Proceedings, Part I*, pages 494–503. Springer, 2022.
7. David Gale. Optimal assignments in an ordered set: an application of matroid theory. *Journal of Combinatorial Theory*, 4(2):176–180, 1968.
8. Romain Guillaume, Adam Kasperski, and Paweł Zieliński. Distributionally robust possibilistic optimization problems. *Fuzzy Sets and Systems*, 454:56–73, 2023.
9. Mikita Hradovich, Adam Kasperski, and Pawel Zielinski. The recoverable robust spanning tree problem with interval costs is polynomially solvable. *Optimization Letters*, 11(1):17–30, 2017.
10. Adam Kasperski. *Discrete optimization with interval data*. Springer, 2008.
11. Adam Kasperski and Paweł Zieliński. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97(5):177–180, 2006.
12. Adam Kasperski and Paweł Zieliński. On combinatorial optimization problems on matroids with uncertain weights. *European Journal of Operational Research*, 177(2):851–864, 2007.
13. Eugene L Lawler. *Combinatorial optimization: networks and matroids*. Courier Corporation, 1976.
14. Arturo I Merino and José A Soto. The minimum cost query problem on matroids with uncertainty areas. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2019.
15. Roberto Montemanni and Luca M. Gambardella. A branch and bound algorithm for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 161(3):771–779, 2005.
16. Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, volume 32. 01 1982.
17. Erik Quaeghebeur, Keivan Shariatmadar, and Gert De Cooman. Constrained optimization problems under uncertainty with coherent lower previsions. *Fuzzy Sets and Systems*, 206:74–88, 2012.

18. Tuan-Anh Vu, Sohaib Afifi, Éric Lefèvre, and Frédéric Pichon. On modelling and solving the shortest path problem with evidential weights. In *Belief Functions: Theory and Applications: 7th International Conference, BELIEF 2022, Paris, France, October 26–28, 2022, Proceedings*, pages 139–149. Springer, 2022.
19. Hande Yaman, Oya Ekin KaraŞan, and Mustafa Ş. Pınar. The robust spanning tree problem with interval data. *Operations Research Letters*, 29(1):31–40, 2001.

A Specific Matroid structures

In this appendix, we recall some specific cases of matroids that can play important roles in some optimisation problems, some of which are discussed in details in Appendix B.

A.1 Transversal matroid

Let E be a finite set of n elements. Let $\mathcal{Q} = (Q_1, Q_2, \ldots, Q_l)$ be a family of subsets of E. It is important to note that the members of the family $\mathcal{Q}$ are *not* necessarily distinct.

A *transversal* (or system of distinct representatives) of $\mathcal{Q}$ is a subset $\{e_1, \ldots e_l\}$ of l elements of E such that for all i, $e_i \in Q_i$. Similarly, a subset $I \subseteq E$ is a *partial* transversal of $\mathcal{Q}$ if for some $K \subseteq \{1, \ldots, l\}$, I is a transversal of the family $(Q_i : i \in K)$.

Definition 6. *$M = (E, \mathcal{I})$ is a transversal matroid with respect to $\mathcal{Q}$ where a subset $I \subseteq E$ is independent, i.e., $I \in \mathcal{I}$ if and only if I is a partial transversal of $\mathcal{Q}$. A base of M is just a transversal of $\mathcal{Q}$.*

There is another definition of transversal matroid, which is used very often in the literature.

Recall that a matching in an undirected graph G is just a set of edges of G without common vertices, i.e., each vertex appears in at most one edge of the matching. Let $G_{\mathcal{Q}} := (\mathcal{Q}, E, A)$ be a bipartie graph where arc $(Q_i, e_j) \in A$ if and only if $e_j \in Q_i$.

Definition 7. *$M = (E, \mathcal{I})$ is a transversal matroid with respect to $\mathcal{Q}$ where a subset $I \subseteq E$ is independent, i.e., $I \in \mathcal{I}$ if and only if I can be matched in $G_{\mathcal{Q}}$.*

It can be easily checked that these definitions are equivalent.

Example 10. Let $E = \{1, 2, 3, 4\}$ and the family $\mathcal{Q} = (Q_1 = \{1, 2, 3, 4\}, Q_2 = \{3, 4\}, Q_3 = \{3, 4\})$. The bipartie graph $G_{\mathcal{Q}} = (\mathcal{Q}, E, A)$ is represented in Figure 10. We see that $\{1, 3, 4\}$ is independent because there exists a matching in $G_{\mathcal{Q}}$ (in blue) that covers it. On the other hand, $\{1, 2\}$ is dependent because it cannot be matched in $G_{\mathcal{Q}}$.

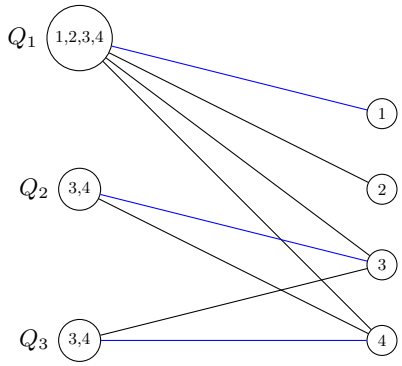

Fig. 7. The bipartite graph of the transversal matroid in Example 10

A.2 Partition matroid

A partition matroid $M = (E, \mathcal{I})$ is a matroid in which the ground set E is partitioned into (disjoint) sets $E_1, E_2, \ldots, E_l$ and

$$\mathcal{I} = \{I \subseteq E : |X \cap E_i| \leq d_i \text{ for all } i = 1, \ldots, l\}, \tag{2}$$

for some given integers $d_i, i = 1, \ldots, l$.

Remark 2. Partition matroid can be seen as a special case of transversal matroid: take the family $\mathcal{Q}$ in the Definition 7 such that $\mathcal{Q}$ contains each set E_i exactly d_i times.

B Some practical matroid optimization problems

B.1 The task scheduling problem

There is a set E of n tasks to be completed one by one by a single machine. The required time for each task is one unit time. Each task j has a deadline $1 \leq d_j \leq n$, the final time by which it must be completed. A penalty w_j incurs if the task is not finished by its assigned deadline. The goal is to select a schedule (permutation on $\{1, \ldots, n\}$) for tasks in order to minimize the total penalty.

We're going to show that the task scheduling problem, i.e., selecting an optimal (schedule) permutation, can be reduced to the problem of selecting a subset, which is a matroid problem.

Let π be an arbitrary schedule. Denote by $early(\pi), late(\pi)$, the tasks which are early and late under π, respectively. It is obvious that the penalty $W(\pi)$ of the schedule can be computed as:

$$W(\pi) = \sum_{j=1}^{n} w_j - \sum_{j \in early(\pi)} w_j. \tag{3}$$

Job	Deadline	Penalty
1	1	6
2	3	5
3	1	7
4	2	3
5	4	4
6	3	4

Table 4.

Consider a task scheduling problem with data is given in Table 4 and the schedule $\pi = 1 \to 2 \to 3 \to 4 \to 5 \to 6$. Under this schedule, the early tasks are $early(\pi) = (1, 2)$ and the late tasks are $late(\pi) = (3, 4, 5, 6)$.

We call a set of tasks I is on-time if there exists a schedule for tasks in I such that no task is late.

Example 11. The set $I = \{1, 2\}$ is on-time because in the schedule $1 \to 2$ no task is late. Note that in the schedule $2 \to 1$, task 1 is late.

Note 1. I is on-time if and only if when tasks in I are scheduled in increasing order of deadlines no task is late.

Proof. The direction ($\Leftarrow$) is obvious. For the other direction, let π_I be a schedule such that under π_I no task is late. Assume there exists two consecutive tasks $i \to j$ in π_I with the finishing times $t, t+1$, respectively and $d_j < d_i$. Because j is early so $t + 1 \leq d_j$. Now swap positions of i and j. Since $d_j < d_i$, so $t + 1 < d_i$. Hence, after the swap, i is early. Finally, j is still early because it is now performed even earlier. Repeat such swap to eventually get a new on-time schedule for I in which tasks are scheduled in increasing order of deadlines. □

Note 2. The task scheduling problem boils down to finding an on-time set of tasks with maximum total penalty.

Proof. Let π be a schedule with minimum total penalty, then thanks to Equation 3, the set of tasks $early(\pi)$ is the one with maximum total penalty. Conversely, let I be an on-time set with with maximum total penalty. Create a schedule by executing tasks in I in increasing order of deadlines while the rest of tasks can be performed under any order. Again, thanks to Equation 3, this schedule has minimum penalty. □

The problem is now reduced to finding an on-time set with maximum total penalty.

Note 3. The pair $M = (E, \mathcal{I})$ is a transversal matroid where $I \in \mathcal{I}$ if and only if I is on-time

To see this, let a family $\mathcal{Q} = (Q_t : t = 1, \ldots, n)$ such that $Q_t = \{j \in E : d_j \geq t\}$. In other words, Q_t is a set of tasks whose deadline are at least t. We show that M is indeed a transversal matroid problem with respect to $\mathcal{Q}$.

Proof. If I is on-time then by Note 1, no task is late when tasks in I are scheduled in increasing order of deadlines. Let $j_1 \to \ldots \to j_m$ is such schedule of I with $d_{j_1} \leq \ldots \leq d_{j_m}$. Because no task is late so the deadlines $d_{j_1} \geq 1, \ldots, d_{j_m} \geq m$. It follows directly from the definition of the family $\mathcal{Q}$ that I is indeed a partial transversal of $\mathcal{Q}$: $j_1 \in Q_1, \ldots, j_m \in Q_m$.

Conversely, if I is not on-time then there is a late task j_k in the schedule (of increasing order of deadlines) $j_1 \to \ldots j_k \ldots \to j_m$, and thus its deadline $d_{j_k} \leq k-1$. Hence, $d_{j_1} \leq \ldots \leq d_{j_k} \leq k-1$. It follows directly (the pigeonhole principle) that we cannot match k tasks $\{j_1, \ldots, j_k\}$ into $k-1$ sets $Q_1, \ldots, Q_{k-1}$. So I is not a partial transversal of $\mathcal{Q}$.

□

Example 12. In the concrete problem with data given as in Table 4, we have $Q_1 = \{1,2,3,4,5,6\}, Q_2 = \{2,4,5,6\}, Q_3 = \{2,5,6\}$, and $Q_4 = \{5\}$. Let us construct the bipartie graph $G_{\mathcal{Q}} = (\mathcal{Q}, E, A)$ as described in Section A.1. $G_{\mathcal{Q}}$ is given in Figure 8. Note that $Q_5 = Q_6 = \varnothing$, so they are discarded from the graph.

The set $I = \{1,4,2,5\}$ is on-time and the corresponding matching is depicted in blue.

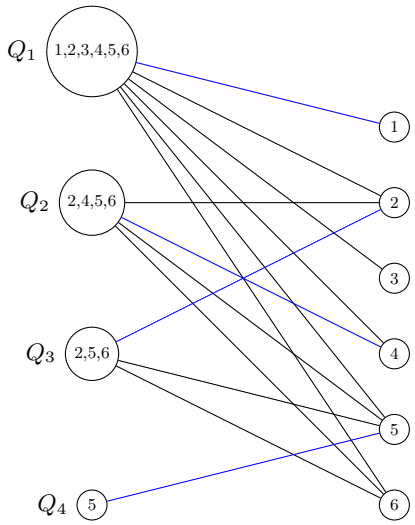

Fig. 8. The bipartite graph in Example 12

Having shown that the problem is a matroid optimization problem, we can solve it as follows.

– Sort tasks in decreasing order of penalties.

– Start with the empty set and add tasks one at a time as long as the resulting set of tasks is on-time.
– Eventually, we'll obtain an on-time set with maximum total penalty. Then we create an optimal schedule (permutation) with minimum total penalty as described in the proof of Note 2.

To implement the greedy algorithm, we need a procedure to check if a given set I of m tasks ($1 \leq m \leq n$) is on-time (independent). By Note 1, an obvious approach is to sort tasks in I in increasing order of deadlines $j_1 \rightarrow, \ldots, \rightarrow j_m$ and check if $d_{j_1} \geq 1, \ldots, d_{j_m} \geq m$. This procedure takes $O(n \log n)$.

A faster procedure, without the need for sorting, is as follows. Note that the second part of the proof of Note 3 already shows that if there are k tasks in I with deadlines at most $k-1$, then I is definitely *not* on-time. With only minor arguments, it follows that

$$I \text{ is on-time} \Leftrightarrow I_t \leq t \ \forall t = 1, \ldots, n, \tag{4}$$

where I_t is the number of tasks in I whose deadline are at most t, i.e., $I_t = |\{j \in I : d_j \leq t\}|$. Using (4), a new test based on counting tasks in I (Algorithm 8) runs in $O(n)$. Overall, the greedy algorithm runs in $O(n^2)$.

Example 13. Consider again the problem given in Table 4. Because two tasks 1 and 3 have deadline 1, so any set that contains 1 and 3 is not (independent) on-time.

Algorithm 8: Check if I is on-time (independent) or not

Data: a set of tasks I
Result: Whether I is on-time or not

```
1  A ← [0]*n, V ← [0]*n ;                    // Initialize arrays of n zeros
2  for j in I do
3  |  A[j.deadline] ← A[j.deadline] + 1 ;     // Save number of tasks in I
   |                                          // with deadline j.deadline
4  V[1] ← A[1];
5  if V[1] > 1 then
6  |  return not on time;
7  for t in 2, ..., n do
8  |  V[t] ← V[t − 1] + A[t] ;    // Number of tasks in I with deadline ≤ t
9  |  if V[t] > t then
10 |  |  return not on time;
11 return on time;
```

B.2 The semi matching problem

Let W be an $m \times n$ matrix, where each entry has a weight w_{ij}. The problem is to choose a set of entries with maximum weight in W such that no two entries are from the same row of W.

Let the ground set E consists of $m \times n$ entries of W. We'll show that the pair $M = (E, \mathcal{I})$ is a (partition) matroid where $I \in \mathcal{I}$ if and only if no two entries in I are from the same row.

Proof. Recall the definition of partition matroid in Section A.2. Let a partition of E be such that $E = E_1 \cup \ldots \cup E_m$ where E_i is just the ith row of the matrix. By the definition, M is a partition matroid with respect to $E_1, \ldots, E_m$. □

Hence, the greedy algorithm works as follows: start with the empty set and add entries one at a time in decreasing order of weights, as long as no two entries in the resulting set are from the same row. The implementation for the independence test is evident.

B.3 The assignment problem

There is a set of n jobs E, each job j has a weight w_j that represents the profit if it is performed. There is a set of m agents A. Each agent x is able to perform a set of jobs E_x. The goal is to select a set of jobs to be performed for maximum profit. The requirement is that two different jobs are assigned to two different agents. Note that not all jobs are assigned. This is normal, because in some cases there are more jobs than agents.

By definition of transversal matroid in Section A.1, the problem is indeed a (transversal) matroid optimization problem.

Example 14. Consider a problem with data given in Table 5 where, for instance, agent 1 can do jobs 1,2,3 and 4. The corresponding bipartite G is given in Figure 9.

Agent	Doable jobs
1	{1,2,3,4}
2	{2,5,6}
3	{1,4,6}
4	{3,5}

Table 5.

Hence, the greedy algorithm can be applied by choosing jobs in decreasing order of profits. To check if a set in the current iteration, say, $I = \{1, 2, 4\}$ is independent amounts to checking if I can be matched in G. It can be done, for example, by considering the subgraph G_I of G (depicted in Figure 10) and checking if I is the maximum cardinality matching of G_I, for which there exists standard algorithms.

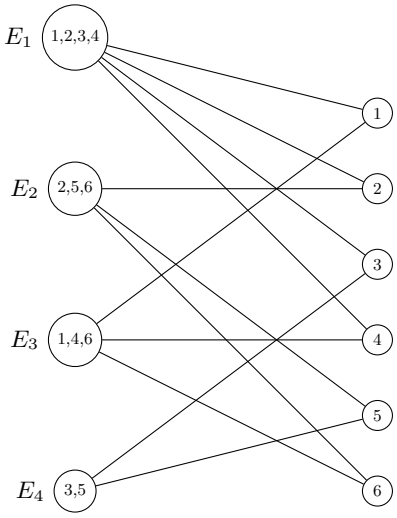

Fig. 9. The bipartite G in Example 14

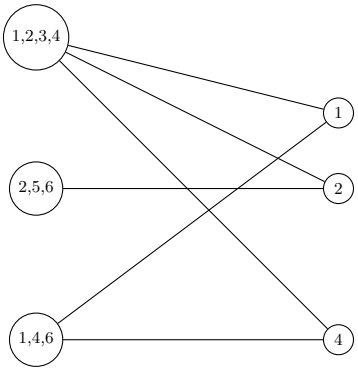

Fig. 10. The subgraph G_I