Polynomial-Delay Enumeration of Large Maximal Common Independent Sets in Two Matroids

Yasuaki Kobayashi
Hokkaido University, Sapporo, Japan

Kazuhiro Kurita
Nagoya University, Nagoya, Japan

Kunihiro Wasa
Hosei University, Tokyo, Japan

Abstract

Finding a *maximum* cardinality common independent set in two matroids (also known as Matroid Intersection) is a classical combinatorial optimization problem, which generalizes several well-known problems, such as finding a maximum bipartite matching, a maximum colorful forest, and an arborescence in directed graphs. Enumerating all *maximal* common independent sets in two (or more) matroids is a classical enumeration problem. In this paper, we address an "intersection" of these problems: Given two matroids and a threshold τ, the goal is to enumerate all maximal common independent sets in the matroids with cardinality at least τ. We show that this problem can be solved in polynomial delay and polynomial space. We also discuss how to enumerate all maximal common independent sets of two matroids in non-increasing order of their cardinalities.

2012 ACM Subject Classification Mathematics of computing → Enumeration

Keywords and phrases Polynomial-delay enumeration, Ranked Enumeration, Matroid intersection, Reverse search

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.58

Related Version *Full Version*: https://arxiv.org/abs/2307.08948

Funding *Yasuaki Kobayashi*: JSPS KAKENHI Grant Numbers JP20H00595 and JP23H03344
Kazuhiro Kurita: JP21K17812, JP22H03549, JST CREST Grant Number JPMJCR18K3, and JST ACT-X Grant Number JPMJAX2105
Kunihiro Wasa: JSPS KAKENHI Grant Numbers JP22K17849 and JP22H03549, and JST CREST Grant Number JPMJCR18K3

1 Introduction

The bipartite matching problem is arguably one of the most famous combinatorial optimization problems, which asks to find a maximum cardinality matching in a bipartite graph. By polynomial-time algorithms for the maximum flow problems, this problem can be solved in polynomial time. This problem is naturally generalized for non-bipartite graphs, which is also solvable in polynomial time [9].

Another natural generalization of the bipartite matching problem is Matroid Intersection. In this problem, we are given two matroids $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$, where $\mathcal{I}_1 \subseteq 2^S$ and $\mathcal{I}_2 \subseteq 2^S$ are the set of independent sets of M_1 and M_2, respectively, and asked to find a maximum cardinality *common independent set* of M_1 and M_2, that is, a maximum cardinality set in $\mathcal{I}_1 \cap \mathcal{I}_2$. When both M_1 and M_2 are partition matroids, this problem is equivalent to the bipartite b-matching problem, which is a generalization of the bipartite matching problem. A famous matroid intersection theorem [10] shows a min-max formula and also gives a polynomial-time algorithm for Matroid Intersection [23].

48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023).
Editors: Jérôme Leroux, Sylvain Lombardy, and David Peleg; Article No. 58; pp. 58:1–58:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

These classical results give efficient algorithms to find a *single* best (bipartite) matching in a graph or common independent set in two matroids. This type of objective serves as the gold standard in many algorithmic and computational studies. However, such a single best solution may not be appropriate for real-world problems due to the complex nature of them [11].

One possible remedy to this issue is to enumerate *multiple* solutions instead of a single best one. From the point of view of enumeration, the problems of enumeration *maximal* and *maximum* (bipartite) matchings and its generalization are studied in the literature [6, 29–31]. Enumerating maximal independent sets in a graph is one of the best-studied problems in this area and is solvable in polynomial delay and polynomial space [6, 29]. Due to the correspondence between matchings in a graph and independent sets in its line graph, we can enumerate all maximal matchings in polynomial delay and polynomial space as well. Moreover, several algorithms that are specialized to (bipartite) matchings are known [30, 31].

Enumeration algorithms for matroids are also frequently studied in the literature [3, 12, 16, 17, 22]. Lawler et al. [22] showed that all maximal common independent sets in k matroids can be enumerated in polynomial delay when k is constant. For general k, this problem is highly related to Dualization (or equivalently, minimal transversal enumeration, minimal hitting set enumeration), which can be solved in output quasi-polynomial time[1] [3]. Apart from common independent sets, enumeration problems related to matroids are studied [15, 17], such as minimal multiway cuts [17] and minimal Steiner forests in graphs [15].

In this paper, we consider an "intersection" of the above two worlds, optimization and enumeration, for Matroid Intersection. More specifically, given two matroids M_1 and M_2 and an integer τ, we consider the problem of enumerating all *maximal* common independent sets of M_1 and M_2 *with cardinality at least* τ. We refer to this problem as Large Maximal Common Independent Set Enumeration. By setting $\tau = 0$, we can enumerate all maximal common independent sets of M_1 and M_2, and by setting $\tau = \mathrm{opt}$, we can enumerate all maximum common independent sets of M_1 and M_2, where opt is the optimal value of Matroid Intersection. We would like to mention that simultaneously handling two constraints, maximality and cardinality, would make enumeration problems more difficult (see [18–20], for other enumeration problems). We show that Large Maximal Common Independent Set Enumeration can be solved in polynomial delay and space. This extends the results of enumerating maximum common independent sets due to [12] and enumerating maximal common independent sets due to [22]. Our enumeration algorithm allows us to enumerate several combinatorial objects with maximality and cardinality constraints, such as bipartite b-matchings, colorful forests, and degree-constraint subdigraphs.

To prove this, we devise a reverse search algorithm [1] to enumerate all maximal common independent sets of M_1 and M_2. This algorithm enumerates the solutions in a depth first manner. To completely enumerate all the solutions without duplicates, we carefully design its search strategy. We exploit a famous augmenting path theorem for Matroid Intersection [23]. This enables us to design a "monotone" search strategy, yielding a polynomial-delay and polynomial-space enumeration algorithm. A similar idea is used in [19] for enumerating maximal matchings with cardinality at least τ but we need several nontrivial lemmas to obtain our result.

Although our algorithm enumerates all maximal common independent sets of two matroids with cardinality at least τ, solutions may not be generated in a sorted order, which is of great importance in database community [8, 26]. A *ranked enumeration algorithm* is an algorithm

---

[1] An enumeration algorithm runs in output quasi-polynomial time if it runs in time $N^{(\log N)^c}$, where c is a constant and N is the combined size of the input and output.

enumerates all the solutions in a non-increasing order of their cardinality (or more generally objective value). We discuss how to convert our enumeration algorithm to the one that enumerates in a ranked manner with a small overhead in the running time.

2 Preliminaries

Let S be a finite set. We denote the cardinality of S as n. For two sets X and Y, the symmetric difference of X and Y is defined as $X \triangle Y := (X \setminus Y) \cup (Y \setminus X)$. A pair $M = (S, \mathcal{I})$ is called a *matroid* if M satisfies the following conditions:

- $\emptyset \in \mathcal{I}$,
- if $I \in \mathcal{I}$ and $J \subseteq I$, then $J \in \mathcal{I}$, and
- if $I, J \in \mathcal{I}$ and $|I| < |J|$, then $I \cup \{e\} \in \mathcal{I}$ for some $e \in J \setminus I$.

A subset S' of S is called an *independent set* of M (or *independent* in M) if S' is contained in $\mathcal{I}$ and S' is called a *dependent set* of M (or *dependent* in M) otherwise. An inclusion-wise maximal independent set of M is called a *base* of M, and an inclusion-wise minimal dependent set of M is called a *circuit* of M. For two distinct circuits C_1 and C_2 of M with $C_1 \cap C_2 \neq \emptyset$ and $e \in C_1 \cap C_2$, there always exists a circuit C_3 of M such that $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$. This property is called the (*weak*) *circuit elimination axiom* [25]. For a matroid $M = (S, \mathcal{I})$ and a subset $X \subseteq S$, the pair $(X, \mathcal{J})$ is the *restriction* of M to X, where $\mathcal{J} = \{Y \subseteq X : Y \in \mathcal{I}\}$. We denote it as $M \mid X$. Similarly, the pair $(S \setminus X, \mathcal{J}')$ is the *deletion* of X from M, where $\mathcal{J}' = \{Y \setminus X : Y \in \mathcal{I}\}$. We denote it as $M \setminus X$. Moreover, the pair $(S \setminus X, \mathcal{J}'')$ is the *contraction* of X from M, where $\mathcal{J}'' = \{Y \subseteq S \setminus X : M \mid X \text{ has a base } B \text{ such that } Y \cup B \in \mathcal{I}\}$. We denote it as M/X. Similarly, It is known that for a matroid $M = (S, \mathcal{I})$ and $X \subseteq S$, M/X, $M \mid X$, and $M \setminus X$ are all matroids [25]. For two matroids $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ defined on the same set S, a subset $T \subseteq S$ is a *common independent set* of M_1 and M_2 if $T \in \mathcal{I}_1$ and $T \in \mathcal{I}_2$.

Let I_1 and I_2 be distinct independent sets of M. In our algorithm, we frequently consider a matroid obtained from M by restricting to $I_1 \cup I_2$ and then contracting $I_1 \cap I_2$. This matroid is defined on $I_1 \triangle I_2$ and has some properties shown below.

▶ **Proposition 1.** *Let I_1 and I_2 be independent sets of M and let $M' = (M|(I_1 \cup I_2))/(I_1 \cap I_2)$. $I \subseteq I_1 \triangle I_2$ is independent in M' if and only if $I \cup (I_1 \cap I_2)$ is independent in M.*

Proof. Suppose that I is independent in M'. As M' is a contraction of $I_1 \cap I_2$ from $M|(I_1 \cup I_2)$, $I \cup (I_1 \cap I_2)$ is independent in $M \mid (I_1 \cup I_2)$ and hence in M. The converse direction is analogous. ◀

▶ **Proposition 2.** *Let I_1 and I_2 be maximal common independent sets of $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$. Then, both $I_1 \setminus I_2$ and $I_2 \setminus I_1$ are maximal common independent sets of two matroids $M'_1 = (M_1 \mid (I_1 \cup I_2)) / (I_1 \cap I_2)$ and $M'_2 = (M_2 \mid (I_1 \cup I_2)) / (I_1 \cap I_2)$.*

Proof. By symmetry, it suffices to show that $I_1 \setminus I_2$ is a maximal common independent set of M'_1 and M'_2. By Proposition 1, $(I_1 \setminus I_2) \cup (I_1 \cap I_2) = I_1$ is independent in M_1 if and only if $I_1 \setminus I_2$ is independent in M'_1. Similarly, I_1 is independent in M_2 if and only if $I_1 \setminus I_2$ is independent in M'_2. Thus, $I_1 \setminus I_2$ is a common independent set of M'_1 and M'_2. To see the maximality, suppose that there is $e \in I_2 \setminus I_1$ such that $(I_1 \setminus I_2) \cup \{e\}$ is a common independent set of M'_1 and M'_2. By Proposition 1, $I_1 \cup \{e\}$ is a common independent set of M_1 and M_2, contradicting the maximality of I_1. ◀

We next define some notations for directed graphs. In this paper, we assume that directed graphs have no self-loops. For a directed graph $D = (V, A)$, we say that a vertex v is an *out-neighbor* of u (u is an *in-neighbor* of v) in D if D has an arc (u, v). The set of out-neighbors of v is denoted by $N^+(v)$, and the set of in-neighbors of v is denoted by $N^-(v)$. A sequence $(v_1, \ldots, v_k)$ of distinct vertices is a *directed path* if there is an arc (v_i, v_{i+1}) for $1 \le i < k$. A directed path $(v_1, \ldots, v_k)$ in D is called a *directed path without shortcuts* if D has no arc from v_i to v_j for any $1 \le i < j \le k$ with $i + 1 < j$.

We measure the time complexity of enumeration algorithms with delay complexity [13]. The *delay* of an enumeration algorithm is the maximum time elapsed between two consecutive outputs, including preprocessing and post-processing time. An enumeration algorithm is called a *polynomial-delay enumeration algorithm* if its delay is upper bounded by a polynomial of the size of an input. An enumeration algorithm is called an *linear incremental-time enumeration algorithm* if, for any $i \le N$, an algorithm outputs at least i solutions in time $O(i \cdot \mathrm{poly}(n))$, where N is the number of solutions [4].

Now, we formally define our problems. Throughout the paper, we assume that matroids are given as *independence oracles*, that is, for a matroid $M = (S, \mathcal{I})$, we can test whether a subset $X \subseteq S$ belongs to $\mathcal{I}$ by accessing an oracle for M. Moreover, we assume that independence oracles can be evaluated in Q time and in $\hat{Q}$ space. We say that an enumeration algorithm runs in polynomial delay (resp. polynomial space) if the delay (resp, space) is upper bounded by a polynomial in $n + Q$ (resp. $n + \hat{Q}$).

▶ **Definition 3.** *Given two matroids $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ represented by independence oracles and an integer τ, Large Maximal Common Independent Set Enumeration asks to enumerate all maximal common independent sets of M_1 and M_2 with cardinality at least τ.*

▶ **Definition 4.** *Given two matroids $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ represented by independence oracles, Ranked Maximal Common Independent Set Enumeration asks to enumerate all maximal common independent sets of M_1 and M_2 in a non-increasing order with respect to cardinality.*

2.1 Overview of an algorithm for finding a maximum common independent set

Our proposed algorithm for Large Maximal Common Independent Set Enumeration leverages a well-known property used in an algorithm for finding a maximum common independent set in two matroids. In this paper, we refer to a particular algorithm given by Lawler [23].

Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be matroids. In Lawler's algorithm [23], we start with an arbitrary common independent set I of M_1 and M_2 (e.g., $I := \emptyset$), update I to a larger common independent set I' in some "greedy way". This update procedure is based on the following auxiliary directed graph $D_{M_1,M_2}(I) = (S \cup \{s, t\}, A)$.

Let $I \subseteq S$ be a common independent set of M_1 and M_2. The set $A = A_1 \cup A_2 \cup A_3 \cup A_4$ of arcs in $D_{M_1,M_2}(I)$ consists of the following four types of arcs. The first type of arcs is defined as

$$A_1 = \{(e, f) \colon e \in I, f \in S \setminus I, I \cup \{f\} \notin \mathcal{I}_1, (I \cup \{f\}) \setminus \{e\} \in \mathcal{I}_1\},$$

that is, an arc $(e, f) \in A_1$ indicates that $I \triangle \{e, f\}$ is independent in M_1. Symmetrically, the second type of arcs is defined as

$$A_2 = \{(f, e) \colon e \in I, f \in S \setminus I, I \cup \{f\} \notin \mathcal{I}_2, (I \cup \{f\}) \setminus \{e\} \in \mathcal{I}_2\},$$

that is, an arc (f, e) indicates that $I \triangle \{e, f\}$ is independent in M_2. The third and fourth types of arcs are defined as

$$A_3 = \{(s, f) \colon f \in S \setminus I, I \cup \{f\} \in \mathcal{I}_1\}$$
$$A_4 = \{(f, t) \colon f \in S \setminus I, I \cup \{f\} \in \mathcal{I}_2\},$$

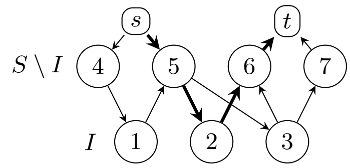

Figure 1 This figure depicts an example of the auxiliary graph $D_{M_1,M_2}(\{1,2,3\})$. Let M_1 and M_2 be matroids with the same ground set $\{1,\ldots,7\}$ that defined by five bases $\{1,2,3,4\}$, $\{1,2,3,5\}$, $\{1,3,5,6\}$, $\{1,2,5,6\}$, $\{1,2,5,7\}$ and six bases $\{1,2,3,6\}$, $\{1,2,3,7\}$, $\{1,2,5,6\}$, $\{1,3,5,6\}$, $\{1,2,5,7\}$, $\{2,3,4,6\}$, respectively. In this example, $D_{M_1,M_2}(\{1,2,3\})$ has a directed s-t path $P = (s,5,2,6,t)$ without shortcuts and $\{1,3,5,6\}$ is a common independent set of M_1 and M_2.

respectively. Arcs (s,f) and (f,t) indicate that $I \cup \{f\}$ is independent in M_1 and in M_2, respectively. We illustrate a concrete example of $D_{M_1,M_2}(I)$ in Figure 1. In the following, we simply write $D(I)$ to denote $D_{M_1,M_2}(I)$.

Let P be a directed path from s to t in $D(I)$ without shortcuts. By the definition of $D(I)$, $|V(P) \cap I|$ is one less than $|V(P) \setminus I|$. Moreover, we can prove that $I \triangle (V(P) \setminus \{s,t\})$ is a common independent set of M_1 and M_2 [23], meaning that the common independent set $I \triangle (V(P) \setminus \{s,t\})$ of M_1 and M_2 is strictly larger than I. It is easy to see that $D(I)$ has a directed path from s to t without shortcuts if and only if $D(I)$ has a directed path from s to t. The following lemma summarizes the above discussion and also proves that the converse direction also holds.

▶ **Lemma 5** (Corollary 3.2 in [23]). *Let I be a common independent set in two matroids M_1 and M_2 and $D(I)$ be the auxiliary directed graph. Then, I is a maximum common independent set in M_1 and M_2 if and only if $D(I)$ has no directed s-t path.*

Such a path P in $D(I)$ is called an *augmenting path*. Lemma 5 is helpful to design an algorithm for enumerating all large maximal common independent sets in two matroids.

3 Enumeration of maximum common independent sets

We first consider the problem of enumerating all *maximum* common independent sets of two matroids, which is indeed a special case of Large Maximal Common Independent Set Enumeration, where $\tau = \mathrm{opt}$.[2] It is known that this problem can be solved in amortized polynomial time using the algorithm in [12]. However, an analysis of the delay of this algorithm is not explicitly given in their paper. In order to show an explicit delay bound, we give a polynomial-delay algorithm for Maximum Common Independent Set Enumeration, using a simple flashlight search technique (also known as binary partition and backtracking) [2, 27].

In this technique, an algorithm enumerates solutions by recursively picking one element e in S and partitioning the set of solutions into two subsets; One set consists of solutions including e, and the other set consists of solutions excluding e. After partitioning according to e, the algorithm repeats this partitioning process until all the elements in S are picked. It is easy to see that each solution set obtained in this process contains at most one solution. If a solution set contains a solution, then we just output it. To upper bound the running time of this recursive algorithm, we need to check whether a current solution set is empty

[2] By opt, we mean the maximum cardinality of a common independent set of M_1 and M_2.

in the recursive partition process. We call such a subproblem an *extension problem*. To enumerate maximum common independent sets of matroids, we define Maximum Common Independent Set Extension as follows.

▶ **Definition 6.** *Given two matroids $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$, and two disjoint subsets $In, Ex \subseteq S$. Maximum Common Independent Set Extension asks to find a maximum common independent set I of M_1 and M_2 that satisfies $In \subseteq I$ and $Ex \cap I = \emptyset$.*

In what follows, we call conditions $In \subseteq I$ and $Ex \cap I = \emptyset$ the *inclusion condition* and *exclusion condition*, respectively. Note that for any matroid M, $M \setminus Ex$ and M / In are matroids. The following proposition is straightforward but essential for solving the extension problem.

▶ **Proposition 7.** *Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be two matroids, and In and Ex be disjoint subsets of S. Suppose that In is a common independent set of M_1 and M_2. Let $M'_1 = (M_1 / In) \setminus Ex$ and $M'_2 = (M_2 / In) \setminus Ex$. Then, there is a maximum common independent set I of M_1 and M_2 that satisfies $In \subseteq I$ and $Ex \cap I = \emptyset$ if and only if there is a common independent set I' of M'_1 and M'_2 with the cardinality $|I| - |In|$.*

By the above proposition, we can solve Maximum Common Independent Set Extension in polynomial time by using a polynomial-time algorithm for finding a maximum common independent set of two matroids [28]. Note that by using oracles for M_1 and M_2, we can check whether a subset of S is independent in M'_1 and in M'_2 in time $O(n + Q)$ and space $O(n + \hat{Q})$.

Now, we design a simple flashlight search algorithm, which is sketched as follows. Let $\mathcal{S}(In, Ex)$ be the set of maximum common independent sets of M_1 and M_2 that satisfy both the inclusion and exclusion conditions. Clearly, the set of all maximum common independent sets of M_1 and M_2 corresponds to $\mathcal{S}(\emptyset, \emptyset)$. By solving the extension problem, we can determine whether $\mathcal{S}(In, Ex)$ is empty or not in polynomial time. Moreover, for an element $e \in S \setminus (In \cup Ex)$, $\{\mathcal{S}(In \cup \{e\}, Ex), \mathcal{S}(In, Ex \cup \{e\})\}$ is a partition of $\mathcal{S}(In, Ex)$. Thus, we can enumerate all maximum common independent sets in $\mathcal{S}(In, Ex)$ by recursively enumerating $\mathcal{S}(In \cup \{e\}, Ex)$ and $\mathcal{S}(In, Ex \cup \{e\})$. We give a pseudo-code of our algorithm in Algorithm 1. Finally, we consider the delay of this algorithm. Let $\mathcal{T}$ be a recursion tree defined by the execution of Algorithm 1. As we output a maximum common independent set of M_1 and M_2 at each leaf node in $\mathcal{T}$, the delay of the algorithm is upper bounded by the "distance" of two leaf nodes times the running time required to processing each node in $\mathcal{T}$. The distance between the root and a leaf node of $\mathcal{T}$ is at most n and thus, the distance between two leaf nodes in $\mathcal{T}$ is upper bounded by linear in n. The time complexity of each node in $\mathcal{T}$ is bounded by $O(\mathrm{poly}(n))$. Hence, the delay of Algorithm 1 is polynomial in n. By using an $O(\mathrm{opt}^{3/2} nQ)$-time and $O(n^2 + \hat{Q})$-space algorithm for finding a maximum common independent set of two matroids [7], the following theorem follows.

▶ **Theorem 8.** *We can enumerate all maximum common independent sets of M_1 and M_2 in $O(\mathrm{opt}^{3/2} n^2 Q)$ delay and $O(n^2 + \hat{Q})$ space.*

4 Enumeration of large maximal common independent sets

We propose a polynomial-delay and polynomial-space algorithm for enumerating maximal common independent sets of two matroids M_1 and M_2 with the cardinality at least τ, namely Large Maximal Common Independent Set Enumeration. From Theorem 8, if $\tau = \mathrm{opt}$, we can enumerate all maximum common independent sets of M_1 and M_2 in polynomial delay and polynomial space. Thus, in this section, we assume that $\tau < \mathrm{opt}$.

Algorithm 1 A polynomial-delay and polynomial-space algorithm for enumerating all maximum common independent sets in two matroids.

Procedure `Maximum`($M_1 = (S, \mathcal{I}_1), M_2 = (S, \mathcal{I}_2)$)
  `RecMaximum`($M_1, M_2, \emptyset, \emptyset$)
Procedure `RecMaximum`($M_1, M_2, \mathit{In}, \mathit{Ex}$)
  if $\mathit{In} \cup \mathit{Ex} = S$ **then** Output In, **return**
  Choose an arbitrary $e \in S \setminus (\mathit{In} \cup \mathit{Ex})$
  if *there is a maximum common independent set* I' *of* M_1 *and* M_2 *that satisfies both* $(\mathit{In} \cup \{e\}) \subseteq I'$ *and* $\mathit{Ex} \cap I' = \emptyset$ **then**
    `RecMaximum`($M_1, M_2, \mathit{In} \cup \{e\}, \mathit{Ex}$)
  if *there is a maximum common independent set* I' *of* M_1 *and* M_2 *that satisfies both* $\mathit{In} \subseteq I'$ *and* $(\mathit{Ex} \cup \{e\}) \cap I' = \emptyset$ **then**
    `RecMaximum`($M_1, M_2, \mathit{In}, \mathit{Ex} \cup \{e\}$)

Our proposed algorithm is based on *reverse search* [1], which is one of the frequently used techniques to design efficient enumeration algorithms [14, 19, 21, 24, 29]. One may expect that a flashlight search algorithm similar to that described in the previous section could be designed for Large Maximal Common Independent Set Enumeration, because finding a *maximal* solution is usually easier than finding a *maximum* solution. However, this intuitive phenomenon does not hold for extension problems. In particular, the problem of finding a maximal matching in a bipartite graph that satisfies an exclusion condition is NP-complete [5, 19]. As the set of all matchings in a bipartite graph can be described as the set of common independent sets of matroids, the extension problem for Large Maximal Common Independent Set Enumeration is NP-complete.

Before delving into our algorithm, we briefly sketch an overview of the reverse search technique. Let $\mathcal{S}$ be the set of solutions. In the reverse search technique, we define a set of "special solutions" $\mathcal{R} \subseteq \mathcal{S}$ and a rooted forest (i.e., a set of rooted trees) on $\mathcal{S}$ whose roots belong to $\mathcal{R}$. Suppose that we can enumerate $\mathcal{R}$ efficiently. Then, we can enumerate all solutions in $\mathcal{S}$ by solely traversing the rooted forest from each root solution in $\mathcal{R}$. To this end, for a non-root solution X in $\mathcal{S} \setminus \mathcal{R}$, it suffices to define its parent $\mathtt{par}(X)$ in an appropriate manner. More specifically, to define a rooted forest on $\mathcal{S}$, this parent-child relation must have no cycles. Moreover, to traverse this rooted forest, we need to efficiently enumerate the children of each internal node in the rooted forest.

Now, we turn back to our problem. In the following, we may simply refer to maximal common independent sets of M_1 and M_2 with cardinality at least τ as *solutions*. We define the set of maximum common independent sets of M_1 and M_2 as the root solutions $\mathcal{R}$. We can efficiently enumerate $\mathcal{R}$ by Theorem 8. To define the parent of a solution not in $\mathcal{R}$, fix an arbitrary maximum common independent set R of M_1 and M_2. Let I be a maximal common independent set of M_1 and M_2 with $|I| < |R|$. We consider two matroids $M_1(R, I) \coloneqq (M_1 \mid (R \cup I)) \,/\, (R \cap I)$ and $M_2(R, I) \coloneqq (M_2 \mid (R \cup I)) \,/\, (R \cap I)$ as well as the auxiliary directed graph $D(R, I) \coloneqq D_{M_1(R,I), M_2(R,I)}(I \setminus R)$. Let us note that the vertex set of $D(R, I)$ is $(R \triangle I) \cup \{s, t\}$. Since R and I are independent in both M_1 and M_2, by Proposition 1, $R \setminus I$ and $I \setminus R$ are independent in $M_1(R, I)$ and $M_2(R, I)$ as well. Moreover, as $|R| > |I|$, we have $|R \setminus I| > |I \setminus R|$. Thus, by Lemma 5, $D(R, I)$ has a directed s-t path. Let P be a directed s-t path $(v_1 = s, v_2, \ldots, v_{2k+1} = t)$ in $D(R, I)$ without shortcuts. We first show that $I \triangle \{v_2, v_3\}$ is a common independent set of M_1 and M_2.

▶ **Lemma 9.** *Let $P = (v_1 = s, \ldots, v_{2k+1} = t)$ be a directed s-t path without shortcuts in $D(R, I)$. Then P has at least four vertices and $I \triangle \{v_2, v_3\}$ is also a common independent set of M_1 and M_2.*

Proof. We first show that P has at least four vertices. As s is not adjacent to t in $D(R, I)$, P has at least three vertices. If $P = (v_1 = s, v_2, v_3 = t)$, then $(I \setminus R) \cup \{v_2\}$ is a common independent set of $M_1(R, I)$ and $M_2(R, I)$. However, by Proposition 2, $I \setminus R$ is a maximal common independent set of $M_1(R, I)$ and $M_2(R, I)$, a contradiction.

We next show that $I \triangle \{v_2, v_3\}$ is also a common independent set of M_1 and M_2. By the definition of $D(R, I)$, $v_2 \in R \setminus I$ and $v_3 \in I \setminus R$. This implies that $v_2 \notin I$ and $v_3 \in I$. Moreover, since $D(R, I)$ has arcs (s, v_2) and (v_2, v_3), $(I \setminus R) \cup \{v_2\}$ is independent in $M_1(R, I)$ and $(I \setminus R) \triangle \{v_2, v_3\}$ is independent in $M_2(R, I)$. By Proposition 1, $I \cup \{v_2\}$ and $I \triangle \{v_2, v_3\}$ are independent in M_1 and in M_2, respectively. As $I \triangle \{v_2, v_3\} = (I \setminus \{v_3\}) \cup \{v_2\}$ is a subset of $I \cup \{v_2\}$, $I \triangle \{v_2, v_3\}$ is a common independent set of M_1 and M_2. ◀

We define the parent $\texttt{par}(I)$ of I as follows. To ensure the consistency of defining its parent, we choose a path P from s to t without shortcuts in $D(R, I)$ in a certain way, and hence the path P is determined solely by the pair R and I. We define the parent of I (under R) as $\mu(I \triangle \{v_2, v_3\})$, where $\mu(X)$ is an arbitrary maximal common independent set of M_1 and M_2 containing X for a common independent set X of M_1 and M_2. Similarly, we choose a maximal common independent set $\mu(X)$ in a certain way, and hence $\mu(X)$ is determined solely by X. Therefore, by Lemmas 5 and 9, for a maximal common independent set I of M_1 and M_2 with $|I| < |R|$, the parent of I (under R) is uniquely determined and we denote it as $\texttt{par}(I)$. In the following, we claim that this parent-child relation defines a rooted forest on the solutions whose roots belong to $\mathcal{R}$. A key to this is a certain "monotonicity", which will be proven in Lemma 11: For any solution I with $|I| < |R|$, it holds that $|R \triangle I| > |R \triangle \texttt{par}(I)|$. Given these, from any solution I with $|I| < |R|$, we can "reach" a maximum common independent set of M_1 and M_2 (not necessarily to be R) by iteratively taking its parent at most n times as $|R \triangle I| \leq n$. To show this monotonicity, we give the following technical lemma, whose proof is deferred to the end of this section.

▶ **Lemma 10.** *Let I be a maximal common independent set of M_1 and M_2 with $|I| < |R|$ and $e \in I$ and $f \in S \setminus I$. If $D(I)$ has two arcs (s, f) and (f, e), then $I \triangle \{e, f\}$ is a common independent set of M_1 and M_2. Moreover, $|\mu(I \triangle \{e, f\})| \leq |I| + 1$.*

Now, we prove the aforementioned "monotonicity".

▶ **Lemma 11.** *Let I be a maximal common independent set of M_1 and M_2 with $|I| < |R|$. Then, the following three properties on $\texttt{par}(I)$ are satisfied.*

- $|I| \leq |\texttt{par}(I)|$,
- $|R \triangle I| > |R \triangle \texttt{par}(I)|$, *and*
- $|I \triangle \texttt{par}(I)| \leq 3$.

Proof. As $|R \setminus I| > |I \setminus R|$, by Lemma 5, there is a directed path $P = (v_1, \ldots, v_{2k+1})$ from $s = v_1$ to $t = v_{2k+1}$ without shortcuts in $D(R, I)$. By Lemma 9, $I' = I \triangle \{v_2, v_3\}$ is a common independent set of M_1 and M_2. The first property $|I| \leq |\texttt{par}(I)|$ follows from

$$|\texttt{par}(I)| = |\mu(I \triangle \{v_2, v_3\})| \geq |I \triangle \{v_2, v_3\}| = |I|,$$

as $v_2 \notin I$ and $v_3 \in I$. Since $v_2 \in R \setminus I$ and $v_3 \in I \setminus R$, we have $|R \triangle I'| = |R \triangle I| - 2$. If $D(I)$ has arcs (s, v_2) and (v_2, v_3), by Lemma 10, $|\mu(I')| \leq |I| + 1$, which yields that

$$|R \triangle \texttt{par}(I)| = |R \triangle \mu(I')| \leq |R \triangle I'| + 1 = |R \triangle I| - 1,$$

where the inequality $|R \triangle \mu(I')| \leq |R \triangle I'| + 1$ follows from $|\mu(I')| \leq |I| + 1 = |I'| + 1$, meaning that $\mu(I') \setminus I'$ contains at most one element. This also shows that

$$|I \triangle \texttt{par}(I)| = |I \triangle \mu(I')| \leq |I \triangle I'| + 1 \leq |I \triangle (I \triangle \{v_2, v_3\})| + 1 = 3.$$

Thus, it suffices to show that $D(I)$ has these arcs (v_2, v_3) and (s, v_2). Let $M_1(R, I) = (M_1 \mid (R \cup I)) \,/\, (R \cap I)$ and $M_2(R, I) = (M_2 \mid (R \cup I)) \,/\, (R \cap I)$. As $D(R, I)$ has the arc (v_2, v_3), $(I \setminus R) \cup \{v_2\}$ and $((I \setminus R) \cup \{v_2\}) \setminus \{v_3\}$ are dependent and independent in $M_2(R, I)$, respectively. By Proposition 1,

$$((I \setminus R) \cup \{v_2\}) \cup (R \cap I) = I \cup \{v_2\} \text{ and}$$
$$(((I \setminus R) \cup \{v_2\}) \setminus \{v_3\}) \cup (R \cap I) = (I \cup \{v_2\}) \setminus \{v_3\}$$

are dependent and independent in M_2, respectively. This implies that $D(I)$ has an arc (v_2, v_3). A similar argument for (s, v_2) and $M_1(R, I)$ proves that $D(I)$ has arc (s, v_2), completing the proof of this lemma. ◀

Now, we are ready to describe our algorithm, which is also shown in Algorithm 2. We assume that the size of a maximum common independent set of M_1 and M_2 is at least τ as otherwise we do nothing. We first enumerate the set $\mathcal{R}$ all maximum common independent sets of M_1 and M_2. This can be done in polynomial delay and polynomial space using the algorithm in Theorem 8. We choose an arbitrary $R \in \mathcal{R}$ and for each $I \in \mathcal{R}$, we enumerate all solutions that belong to the component containing I in the rooted forest defined by the parent-child relation. This is done by calling `RecMaximal(`M_1, M_2, I, R, τ`)`. The procedure `RecMaximal(`M_1, M_2, I, R, τ`)` recursively generates solutions I' with $I = \texttt{par}(I')$. We would like to emphasize that the algorithm only generates solutions I' with $|I'| \geq \tau$.

We first claim that all the solutions are generated by this algorithm. To see this, consider an arbitrary solution I. Define a value $v(I)$ as

$$v(I) = \begin{cases} 0 & \text{if } |I| = |R| \\ |I \triangle R| & \text{otherwise.} \end{cases}$$

We prove the claim by induction on $v(I)$. Suppose that $v(I) = 0$. In this case, I is a maximum common independent set of M_1 and M_2, as $|I \triangle R| = 0$ if and only if $I = R$. Then, I is obviously generated as we call `RecMaximal(`M_1, M_2, I, R, τ`)` for all $I \in \mathcal{R}$. Suppose that $v(I) > 0$. Then, I is a maximal common independent set of M_1 and M_2 with $|I| < |R|$. We assume that all the solutions I' with $v(I') = |R \triangle I'| < |R \triangle I|$ is generated by the algorithm. By Lemma 11, we have $|\texttt{par}(I)| \geq |I| \geq \tau$ and $|R \triangle \texttt{par}(I)| < |R \triangle I|$, which implies that $\texttt{par}(I)$ is generated by the algorithm. By the definition of parent, we have $\texttt{par}(I) = \mu(I \triangle \{u, v\})$ and $|\texttt{par}(I) \triangle I| \geq 2$ for some $u, v \in S$. Moreover, by Lemma 11, $|I \triangle \texttt{par}(I)| \leq 3$, the child I of $\texttt{par}(I)$ is computed at line 7. Thus, I is generated by the algorithm as well.

We next claim that all the solutions are generated without duplication. Since we only call `RecMaximal(`M_1, M_2, I', R, τ`)` for $I = \texttt{par}(I')$, it holds that $v(I) < v(I')$. As $v(I) \leq n$ for every solution I, by the uniqueness of the parent of non-maximum solutions, the algorithm generates each solution exactly once. This concludes that the algorithm correctly enumerates all maximal common independent sets of M_1 and M_2 of cardinality at least τ.

Finally, we discuss the running time of the algorithm. We first enumerate all maximum common independent sets of M_1 and M_2. This can be done in time $O(n^{7/2}Q)$ delay. For each solution I, we compute $\texttt{par}(I)$ as follows. We construct the graph $D(R, I)$ that has $n + 2$ nodes and $O(n^2)$ arcs. This can be done by using $O(n^2 Q)$ queries to the oracles for

Algorithm 2 A polynomial-delay and polynomial-space algorithm for enumerating all maximal common independent sets of M_1 and M_2 with the cardinality at least τ.

Procedure `Maximal`$(M_1 = (S, \mathcal{I}_1), M_2 = (S, \mathcal{I}_2), \tau)$
  Choose arbitrary $R \in \mathcal{R}$
  foreach $I \in \mathcal{R}$ **do** `RecMaximal`(M_1, M_2, I, R, τ) `// Use Algorithm 1.`
Procedure `RecMaximal`(M_1, M_2, I, R, τ)
  Output I
  foreach $X \in \binom{S}{3} \cup \binom{S}{2}$ **do**
    $I' \leftarrow I \triangle X$
    if *I' is a maximal common independent set of M_1 and M_2 such that $\tau \leq |I'| < |R|$ and $I = \mathtt{par}(I')$* **then**
      `RecMaximal`(M_1, M_2, I', R, τ)

M_1 and M_2. To find the path P from s to t without shortcuts, we just compute a shortest path from s to t, which can be done in $O(n^2)$ time. Thus, we can compute $I \triangle \{v_2, v_3\}$ in $O(n^2 Q)$ time as well. From $I \triangle \{v_2, v_3\}$, $\mu(I \triangle \{v_2, v_3\})$ can be computed in $O(nQ)$ time. Thus, we can compute $\mathtt{par}(I)$ from I in time $O(n^2 Q)$.

For each call `RecMaximal`(M_1, M_2, I, R, τ), we output exactly one solution. Moreover, the running time of computing all children of I is $O(n^5 Q)$. This can be seen as there are $O(n^3)$ candidates I' of children and we can check in $O(n^2 Q)$ time whether a candidate I' is in fact a child of I. Thus the delay of the algorithm is upper bounded by the time elapsed between two consecutive calls. As the depth of the rooted forest defined by recursive calls is at most n, this can be upper bounded by $O(n^6 Q)$. As for the space complexity, by Theorem 8, we can enumerate all maximum common independent sets of M_1 and M_2 in $O(n^2 + \hat{Q})$ space. In `RecMaximal`, we need to store local variables I and X in each recursive call, which can be done in space $O(n)$. As the depth of the rooted forest is at most n, the space usage for local variables is $O(n^2)$ in total. For each candidate I', we can check in $O(n^2 + \hat{Q})$ whether I' is a maximal common independent set of M_1 and M_2 and whether $I = \mathtt{par}(I')$. Overall, we have the following theorem.

▶ **Theorem 12.** *There is an $O(n^6 Q)$-delay and $O(n^2 + \hat{Q})$-space algorithm for enumerating maximal common independent sets in two matroids with the cardinality at least τ.*

4.1 Proof of Lemma 10

To complete our proof of Theorem 12, we need to show the correctness of Lemma 10. To this end, we focus on $D(I)$. Since I is a maximal common independent set of M_1 and M_2 with $|I| < |R|$, $D(I)$ has a directed s-t path $P = (v_1 = s, v_2 = f, v_3 = e, \ldots, v_{2k+1} = t)$. By the definition of $D(I)$, $I \triangle \{e, f\}$ is a common independent set of M_1 and M_2. We first show that $I \triangle \{e, f\}$ becomes dependent in M_1 when we add an element f' in $N^+_{D(I)}(e)$.

▶ **Lemma 13.** *Let I be a maximal common independent set of M_1 and M_2, e be an element in I, and f_1 and f_2 be distinct two elements in $N^+_{D(I)}(e)$. Then, $I' := (I \setminus \{e\}) \cup \{f_1, f_2\}$ is dependent in M_1.*

Proof. Since $D(I)$ has arcs (e, f_1) and (e, f_2), both $(I \setminus \{e\}) \cup \{f_1\}$ and $(I \setminus \{e\}) \cup \{f_2\}$ are independent in M_1, and $I \cup \{f_1\}$ and $I \cup \{f_2\}$ are dependent in M_1. Thus, M_1 has two circuits C_1 and C_2 that contain $\{e, f_1\}$ and $\{e, f_2\}$, respectively. By the circuit elimination axiom, there is a circuit $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$. Since I' contains $(C_1 \cup C_2) \setminus \{e\}$, it also contains C_3 and hence is dependent in M_1. ◀

▶ **Lemma 14.** *Let I be a maximal common independent set of M_1 and M_2, e be an element in I, and f_1 and f_2 be distinct two elements in $N^-_{D(I)}(e)$. Then, $I' := (I \setminus \{e\}) \cup \{f_1, f_2\}$ is dependent in M_2*

Proof. Since $D(I)$ has arcs (f_1, e) and (f_2, e), both $I \triangle \{e, f_1\}$ and $I \triangle \{e, f_2\}$ are independent in M_2 and $I \cup \{f_1\}$ and $I \cup \{f_2\}$ are dependent in M_2. Thus, M_2 has two circuits C_1 and C_2 that contain $\{e, f_1\}$ and $\{e, f_2\}$, respectively. By the circuit elimination axiom, there is a circuit $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$. Since I' contains $(C_1 \cup C_2) \setminus \{e\}$, it also contains C_3 and hence is dependent in M_2. ◀

We show that $\mu(I \triangle \{e, f\})$ does not contain any element in $S \setminus (I \cup N^+_{D(I)}(e) \cup N^-_{D(I)}(e))$.

▶ **Lemma 15.** *Let I be a maximal common independent set of M_1 and M_2, e be an element in I, and f be an element in $S \setminus (I \cup N^+_{D(I)}(e) \cup N^-_{D(I)}(e))$. Then, $I \triangle \{e, f\}$ is dependent in at least one of M_1 or in M_2.*

Proof. From the maximality of I, $I \cup \{f\}$ is dependent in at least one of M_1 or M_2. Suppose that $I \cup \{f\}$ is dependent on M_2. Then, $I \cup \{f\}$ contains at least one circuit C of M_2 containing f. We show that $I \cup \{f\}$ contains only one circuit of M_2. If $I \cup \{f\}$ contains another circuit C' with $f \in C'$, by the circuit elimination axiom, $(C \cup C') \setminus \{f\}$ contains a circuit, which contradicts the fact that I is independent in M_2. Thus, M_2 has the unique circuit C, which is contained in $I \cup \{f\}$. Observe that $N^+_{D(I)}(f) \cap I = C \setminus \{f\}$, since $(I \cup \{f\}) \setminus \{e'\}$ is independent in M_2 for $e' \in C$ due to the minimality of C. As $f \in S \setminus (I \cup N^+_{D(I)}(e) \cup N^-_{D(I)}(e))$, we have $e \notin C$. Hence, $(I \setminus \{e\}) \cup \{f\}$ contains C, that is, $(I \setminus \{e\}) \cup \{f\}$ is dependent in M_2. When $I \cup \{f\}$ is dependent in M_1, $(I \setminus \{e\}) \cup \{f\}$ is also dependent in M_1 from a similar discussion. ◀

Now we are ready to prove Lemma 10.

▶ **Lemma 10.** *Let I be a maximal common independent set of M_1 and M_2 with $|I| < |R|$ and $e \in I$ and $f \in S \setminus I$. If $D(I)$ has two arcs (s, f) and (f, e), then $I \triangle \{e, f\}$ is a common independent set of M_1 and M_2. Moreover, $|\mu(I \triangle \{e, f\})| \leq |I| + 1$.*

Proof. By the definition of $D(I)$, $I \triangle \{e, f\}$ is a common independent set since $D(I)$ has two arcs (s, f) and (f, e). Thus, $\mu(I \triangle \{e, f\})$ is a maximal common independent set of M_1 and M_2. We show that $|\mu(I \triangle \{e, f\})| \leq |I| + 1$. Since f is contained in $N^-_{D(I)}(e)$, $\mu(I \triangle \{e, f\})$ does not contain elements in $N^-_{D(I)}(e)$ except for f by Lemma 14. Moreover, by Lemma 13, $\mu(I \triangle \{e, f\})$ contains at most one element in $N^+_{D(I)}(e)$. Finally, $\mu(I \triangle \{e, f\})$ does not contain any element in $S \setminus (I \cup N^+_{D(I)}(e) \cup N^-_{D(I)}(e))$ by Lemma 15. Therefore, $\mu(I \triangle \{e, f\}) \setminus (I \triangle \{e, f\})$ contains at most one element. Since $|I \triangle \{e, f\}| = |I|$, $|\mu(I \triangle \{e, f\})| \leq |I| + 1$. ◀

5 Ranked enumeration

In this section, we give a ranked enumeration algorithm for enumerating maximal common independent sets of two matroids $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$. Recall that an enumeration algorithm is called a *ranked enumeration algorithm* if the algorithm enumerates solutions in a non-increasing order of their cardinality. We do this in a slightly general manner.

In what follows, we consider the following abstract problem. Let S be a finite set and $\mathcal{F}$ be a subset of 2^S. Let $\mathcal{A}(\tau)$ be an algorithm that outputs all sets in $\mathcal{F}$ with the cardinality at least τ. We denote the maximum delay complexity and the space complexity from $\mathcal{A}(\tau)$ to $\mathcal{A}(1)$ as $t(n)$ and $s(n)$, respectively. Moreover, we denote the number of outputs of $\mathcal{A}(\tau)$ as $\#\mathcal{A}(\tau)$. Under this problem setting, we construct a ranked enumeration algorithm that outputs the i-th solution in $O(i \cdot n \cdot t(n))$ time with $O(s(n))$ space as follows.

Our idea is simply to execute $\mathcal{A}$ from $\mathcal{A}(n)$ to $\mathcal{A}(1)$. When $\mathcal{A}(k)$ outputs a solution with cardinality more than k, we just ignore it. In other words, when we execute $\mathcal{A}(i)$, all solutions in $\mathcal{F}$ with cardinality exactly i are output. Clearly, we can enumerate all solutions in $\mathcal{F}$ in a non-increasing order of their cardinality. We consider the time and space complexity of this method. It is easy to see that the space complexity of this algorithm is $O(s(n))$ as we just execute $\mathcal{A}$ in order. Thus, we estimate the running time required to output the first i solutions for $i \leq |\mathcal{F}|$. Let $j \geq 1$ be the maximum integer such that $\#\mathcal{A}(j)$ is less than i. Since the delay of $\mathcal{A}$ is bounded by $t(n)$ and $\#\mathcal{A}(j-1)$ is at least i, $\mathcal{A}(j-1)$ outputs the i-th solution in $O(i \cdot t(n))$ time. Since the total running time is bounded by $O((n-j+1) \cdot \#\mathcal{A}(j) \cdot t(n) + i \cdot t(n)) = O(i \cdot n \cdot t(n))$ time, this algorithm outputs the first i solutions in $O(i \cdot n \cdot t(n))$ time.

▶ **Theorem 16.** *Let S be a finite set and $\mathcal{F}$ be a subset of 2^S. For any $1 \leq k \leq \tau$, suppose that we have an algorithm $\mathcal{A}(k)$ that enumerates all sets in $\mathcal{F}$ with the cardinality at least k for any $1 \leq k \leq \tau$ in $t(n)$ delay and $s(n)$ space. Then, there is an algorithm enumerating all subsets in $\mathcal{I}$ in non-increasing order of their cardinality that outputs the first i solutions in $O(i \cdot n \cdot t(n))$ time using $O(s(n))$ space for $i \leq |\mathcal{F}|$.*

We obtain a linear incremental-time and polynomial-space ranked enumeration algorithm for maximal common independent sets of two matroids by combining Theorems 12 and 16.

▶ **Theorem 17.** *There is a linear incremental-time and polynomial-space algorithm for enumerating all maximal common independent sets in two matroids in non-increasing order. This algorithm outputs the first i solutions in $O(i \cdot n^7 Q)$ time.*

6 Applications of our algorithms

Due to an expressive power of Matroid Intersection, Theorems 12 and 17 give enumeration algorithms for various combinatorial objects in a unified way. An example of such objects is to enumerate maximal b-matchings in bipartite graphs. It is known that an intersection of two matroids can represent all objects in the following theorem. See the appendix for details on representing these objects by an intersection of two matroids.

▶ **Theorem 18.** *There are polynomial delay and space enumeration algorithms for*
- *maximal bipartite b-matchings with cardinality at least τ,*
- *maximal colorful forests with cardinality at least τ, and*
- *maximal degree constrained subgraphs in digraphs with cardinality at least τ,*

Moreover, there are linear incremental-time and polynomial-space ranked enumeration algorithms for the above problems.

References

1. D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1):21–46, 1996. doi:10.1016/0166-218X(95)00026-N.
2. Etienne Birmelé, Rui Ferreira, Roberto Grossi, Andrea Marino, Nadia Pisanti, Romeo Rizzi, and Gustavo Sacomoto. Optimal listing of cycles and st-paths in undirected graphs. In *Proceedings of the 2013 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1884–1896. SIAM, 2013. doi:10.1137/1.9781611973105.134.
3. Endre Boros, Khaled Elbassioni, Vladimir Gurvich, and Leonid Khachiyan. Matroid intersections, polymatroid inequalities, and related problems. In Krzysztof Diks and Wojciech Rytter, editors, *Mathematical Foundations of Computer Science 2002*, pages 143–154, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

4 Florent Capelli and Yann Strozecki. Geometric Amortization of Enumeration Algorithms. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science (STACS 2023)*, volume 254 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:22, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.STACS.2023.18`.

5 Katrin Casel, Henning Fernau, Mehdi Khosravian Ghadikolaei, Jérôme Monnot, and Florian Sikora. Extension of some edge graph problems: Standard and parameterized complexity. In *Proc. FCT 2019*, volume 11651 of *Lecture Notes in Computer Science*, pages 185–200. Springer, 2019. `doi:10.1007/978-3-030-25027-0_13`.

6 Alessio Conte, Roberto Grossi, Andrea Marino, Takeaki Uno, and Luca Versari. Listing maximal independent sets with minimal space and bounded delay. In Gabriele Fici, Marinella Sciortino, and Rossano Venturini, editors, *String Processing and Information Retrieval - 24th International Symposium, SPIRE 2017, Palermo, Italy, September 26-29, 2017, Proceedings*, volume 10508 of *Lecture Notes in Computer Science*, pages 144–160. Springer, 2017. `doi:10.1007/978-3-319-67428-5_13`.

7 William H. Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM Journal on Computing*, 15(4):948–957, 1986. `doi:10.1137/0215066`.

8 Shaleen Deep and Paraschos Koutris. Ranked Enumeration of Conjunctive Query Results. In Ke Yi and Zhewei Wei, editors, *24th International Conference on Database Theory (ICDT 2021)*, volume 186 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ICDT.2021.5`.

9 Jack Edmonds. Paths, trees, and flowers. *Can. J. Math.*, 17:449–467, 1965. `doi:10.4153/CJM-1965-045-4`.

10 Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications, (Proceedings, Calgary International Conference on Combinatorial Structures and Their Applications)*, pages 69–87, Calgary, Alberta, 1970.

11 David Eppstein. k-Best Enumeration. In *Encyclopedia of Algorithms*, pages 1003–1006. Springer, New York, NY, 2016. `doi:10.1007/978-1-4939-2864-4_733`.

12 Komei Fukuda and Makoto Namiki. Finding all common bases in two matroids. *Discrete Applied Mathematics*, 56(2):231–243, 1995. Fifth Franco-Japanese Days. `doi:10.1016/0166-218X(94)00088-U`.

13 David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all maximal independent sets. *Inform. Process. Lett.*, 27(3):119–123, 1988. `doi:10.1016/0020-0190(88)90065-8`.

14 Mamadou Moustapha Kanté, Vincent Limouzy, Arnaud Mary, Lhouari Nourine, and Takeaki Uno. Polynomial Delay Algorithm for Listing Minimal Edge Dominating Sets in Graphs. In *Proc. of WADS 2015*, pages 446–457, 2015. `doi:10.1007/978-3-319-21840-3_37`.

15 L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, and K. Makino. Enumerating spanning and connected subsets in graphs and matroids. In Yossi Azar and Thomas Erlebach, editors, *Algorithms – ESA 2006*, pages 444–455, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

16 Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled M. Elbassioni, Vladimir Gurvich, and Kazuhisa Makino. Generating Cut Conjunctions in Graphs and Related Problems. *Algorithmica*, 51(3):239–263, 2008. `doi:10.1007/s00453-007-9111-9`.

17 Leonid G. Khachiyan, Endre Boros, Khaled M. Elbassioni, Vladimir Gurvich, and Kazuhisa Makino. On the complexity of some enumeration problems for matroids. *SIAM J. Discret. Math.*, 19(4):966–984, 2005. `doi:10.1137/S0895480103428338`.

18 Yasuaki Kobayashi, Kazuhiro Kurita, and Kunihiro Wasa. Efficient constant-factor approximate enumeration of minimal subsets for monotone properties with weight constraints. *CoRR*, abs/2009.08830, 2020. `arXiv:2009.08830`.

19 Yasuaki Kobayashi, Kazuhiro Kurita, and Kunihiro Wasa. Polynomial-delay and polynomial-space enumeration of large maximal matchings. In Michael A. Bekos and Michael Kaufmann, editors, *Graph-Theoretic Concepts in Computer Science*, pages 342–355, Cham, 2022. Springer International Publishing.

20 Tuukka Korhonen. Listing small minimal separators of a graph. *CoRR*, abs/2012.09153, 2020. `arXiv:2012.09153`.

21 Kazuhiro Kurita and Yasuaki Kobayashi. Efficient enumerations for minimal multicuts and multiway cuts. In *Proc. MFCS 2020*, pages 60:1–60:14, 2020. `doi:10.4230/LIPIcs.MFCS.2020.60`.

22 E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Generating All Maximal Independent Sets: NP-Hardness and Polynomial-Time Algorithms. *SIAM Journal on Computing*, 9(3):558–565, 1980.

23 Eugene L Lawler. Matroid intersection algorithms. *Mathematical Programming*, 9(1):31–56, 1975. `doi:10.1007/BF01681329`.

24 Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In *Proc. SWAT 2004*, volume 3111 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 2004. `doi:10.1007/978-3-540-27810-8_23`.

25 James Oxley. *Matroid theory*, volume 3. Oxford University Press, USA, 2006.

26 Noam Ravid, Dori Medini, and Benny Kimelfeld. Ranked Enumeration of Minimal Triangulations. In *Proc. of PODS 2019*, pages 74–88, 2019. `doi:10.1145/3294052.3319678`.

27 R. C. Read and R. E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5(3):237–252, 1975. `doi:10.1002/net.1975.5.3.237`.

28 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

29 Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A New Algorithm for Generating All the Maximal Independent Sets. *SIAM J. Comput.*, 6(3):505–517, 1977. `doi:10.1137/0206036`.

30 Takeaki Uno. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. In Hon Wai Leong, Hiroshi Imai, and Sanjay Jain, editors, *Algorithms and Computation*, pages 92–101, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

31 Takeaki Uno. A fast algorithm for enumerating bipartite perfect matchings. In Peter Eades and Tadao Takaoka, editors, *Algorithms and Computation*, pages 367–379, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.