Parameterized Complexity of Conflict-Free Matchings and Paths

Akanksha Agrawal
Ben-Gurion University of the Negev, Beer-Sheva, Israel
agrawal@post.bgu.ac.il

Pallavi Jain
Institute of Mathematical Sciences, HBNI, Chennai, India
pallavij@imsc.res.in

Lawqueen Kanesh
Institute of Mathematical Sciences, HBNI, Chennai, India
lawqueen@imsc.res.in

Saket Saurabh
University of Bergen, Bergen, Norway
Institute of Mathematical Sciences, HBNI, Chennai, India
UMI ReLax
saket@imsc.res.in

Abstract

An input to a conflict-free variant of a classical problem Γ, called Conflict-Free Γ, consists of an instance I of Γ coupled with a graph H, called the *conflict graph*. A solution to Conflict-Free Γ in (I, H) is a solution to I in Γ, which is also an independent set in H. In this paper, we study conflict-free variants of Maximum Matching and Shortest Path, which we call Conflict-Free Matching (CF-Matching) and Conflict-Free Shortest Path (CF-SP), respectively. We show that both CF-Matching and CF-SP are W[1]-hard, when parameterized by the solution size. Moreover, W[1]-hardness for CF-Matching holds even when the input graph where we want to find a matching is itself a matching, and W[1]-hardness for CF-SP holds for conflict graph being a unit-interval graph. Next, we study these problems with restriction on the conflict graphs. We give FPT algorithms for CF-Matching when the conflict graph is chordal. Also, we give FPT algorithms for both CF-Matching and CF-SP, when the conflict graph is d-degenerate. Finally, we design FPT algorithms for variants of CF-Matching and CF-SP, where the conflicting conditions are given by a (representable) matroid.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Fixed parameter tractability; Theory of computation → W hierarchy

Keywords and phrases Conflict-free, Matching, Shortest Path, FPT algorithm, W[1]-hard, Matroids

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.53

1 Introduction

In the recent years, conflict-free variant of classical combinatorial optimization problems have gained attention from the viewpoint of algorithmic complexity. A typical input to a conflict-free variant of a classical problem Γ, which we call Conflict-Free Γ, consists of an instance I of Γ coupled with a graph H, called the *conflict graph*. A solution to Conflict-Free Γ in (I, H) is a solution to I in Γ, which is also an independent set in H. Notice that conflict-free version of the problem introduces the constraint of "impossible pairs" in the solution that we seek for. Such a constraint of "impossible pairs" in a solution arises, for example, in the context of program testing and validation [16, 23]. Gabow et al. [16] studied the conflict-free version of finding paths in a graph, which they showed to be NP-complete.

Conflict-free variants of several classical problems such as, Bin Packing [10, 18, 20], Knapsack [34, 31], Minimum Spanning Tree [5, 6], Maximum Matching [6], Maximum Flow [32, 33], Shortest Path [6] and Set Cover [11] have been studied in the literature from the viewpoint of algorithmic complexity, approximation algorithms, and heuristics. It is interesting to note that most of these problems are NP-hard even when their classical counterparts are polynomial time solvable. Recently, Jain et al. [19] and Agrawal et al. [2, 1] initiated the study of conflict-free problems in the realm of parameterized complexity. In particular, they studied Conflict-Free $\mathcal{F}$-Deletion problems for various families $\mathcal{F}$, of graphs such as, the family of forests, independent sets, bipartite graphs, interval graphs, etc.

Maximum Matching and Shortest Path are among the classical graph problems which are of very high theoretical and practical interest. The Maximum Matching problem takes as input a graph G, and the objective is to compute a maximum sized subset $Y \subseteq E(G)$ such that no two edges in Y have a common vertex. Maximum Matching is known to be solvable in polynomial time [12, 27]. The Shortest Path problem takes as input a graph G and vertices s and t, and the objective is to compute a path between s and t in G with the minimum number of vertices. The Shortest Path problem, together with its variants such as all-pair shortest path, single-source shortest path, weighted shortest path, etc. are known to be solvable in polynomial time [7, 3].

Darmann et al. [6] (among other problems) studied the conflict-free variants of Maximum Matching and Shortest Path. They showed that the conflict-free variant of Maximum Matching is NP-hard even when the conflict graph is a disjoint union of edges (matching). Moreover, for the conflict-free variant of Shortest Path, they showed that the problem is APX-hard, even when the conflict graph belongs to the family of 2-ladders.

In this paper, we study the conflict-free versions of matching and shortest path from the viewpoint of parameterized complexity. A parameterized problem Π is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet. An instance of a parameterized problem is a pair (I, k), where I is a classical problem instance and k is an integer, which is called the *parameter*. One of the central notions in parameterized complexity is *fixed-parameter tractability*, where given an instance (I, k) of a parameterized problem Π, the goal is to design an algorithm that runs in time $f(k)n^{\mathcal{O}(1)}$, where, $n = |I|$ and $f(\cdot)$ is some computable function, whose value depends only on k. An algorithm with running time as described above, is called an FPT algorithm. A parameterized problem that admits an FPT algorithm is said to be in FPT. Not every parameterized problem admits an FPT algorithm, under reasonable complexity-theoretic assumptions. Similar to the notion of NP-hardness and NP-hard reductions in classical Complexity Theory, there are notions of W[t]-hardness, where $t \in \mathbb{N}$ and parameterized reductions in parameterized complexity. A parameterized problem which is W[t]-hard, for some $t \in \mathbb{N}$ is believed not to admit an FPT algorithm. For more details on parameterized complexity we refer to the books of Downey and Fellows [9], Flum and Grohe [13], Niedermeier [29], and Cygan et al. [4].

Our Results. We study conflict-free (parameterized) variants of Maximum Matching and Shortest Path, which we call Conflict Free Maximum Matching (CF-MM, for short) and Conflict Free Shortest Path (CF-SP, for short), respectively. These problems are formally defined below.

CONFLICT FREE MAXIMUM MATCHING (CF-MM)	**Parameter:** k
Input: A graph $G = (V, E)$, a conflict graph $H = (E, E')$, and an integer k.	
Question: Is there a matching M of size at least k in G, such that M is an independent set in H?	

CONFLICT FREE SHORTEST PATH (CF-SP)	**Parameter:** k
Input: A graph $G = (V, E)$, a conflict graph $H = (E, E')$, two special vertices s and t, and an integer k.	
Question: Is there an st-path P of length at most k in G, such that $E(P)$ is an independent set in H?	

We show that both CF-MM and CF-SP are W[1]-hard, when parameterized by the solution size. The W[1]-hardness for CF-MM is obtained by giving an appropriate reduction from INDEPENDENT SET, which is known to be W[1]-hard, when parameterized by the solution size [4, 8]. In fact, our W[1]-hardness result for CF-MM holds even when the graph where we want to compute a matching is itself a matching. We show the W[1]-hardness of CF-SP by giving an appropriate reduction from a multicolored variant of the problem UNIT 2-TRACK INDEPENDENT SET (which we prove to be W[1]-hard). We note that UNIT 2-TRACK INDEPENDENT SET is known to be W[1]-hard, which is used to establish W[1]-hardness of its multicolored variant. We note that our W[1]-hardness result of CF-SP holds even when the conflict graph is a unit interval graph.

Having shown the W[1]-hardness results, we then restrict our attention to having conflict graphs belonging to some families of graphs, where the INDEPENDENT SET problem is either polynomial time solvable or solvable in FPT time. Two of the very well-known graph families that we consider are the family of chordal graphs and the family of d-degenerate graphs. For the CF-MM problem, we give an FPT algorithm, when the conflict graph belongs to the family of chordal graphs. Our algorithm is based on a dynamic programming over a "structured" tree decomposition of the conflict graph (which is chordal) together with "efficient" computation of representative families at each step of our dynamic programming routine. Notice that we cannot obtain an FPT algorithm for the CF-SP problem when the conflict graph is a chordal graph. This holds because unit-interval graphs are chordal, and the problem CF-SP is W[1]-hard, even when the conflict graph is a unit-interval graph.

For conflict graphs being d-degenerate, we obtain FPT algorithms for both CF-MM and CF-SP. These algorithms are based on the computation of an independence covering family, a notion which was recently introduced by Lokshtanov et al. [25]. We note that even for nowhere dense graphs, such an independence covering family can be computed efficiently [25]. Since our algorithms are based on computation of independence covering families, hence, our results hold even when the conflict graph is a nowhere dense graph.

Finally, we study a variant of CF-MM and CF-SP, where instead of conflicting conditions being imposed by independent sets in a conflict graph, they are imposed by independence constraints in a (representable) matroid. We give FPT algorithms for the above variant of both CF-MM and CF-SP.

2 Preliminaries

Sets and functions.

We denote the set of natural numbers and the set of integers by $\mathbb{N}$ and $\mathbb{Z}$, respectively. By $\mathbb{N}_{\geq 1}$ we denote the set $\{x \in \mathbb{N} \mid x \geq 1\}$. For $n \in \mathbb{N}$, by $[n]$ and $[0, n]$, we denote the sets $\{1, 2, \cdots, n\}$ and $\{0, 1, 2, \cdots, n\}$, respectively. For a set U and $p \in \mathbb{N}$, a p-family (over U)

is a family of subsets of U of size p. A function $f : X \to Y$ is *injective* if for each $x, y \in X$, $f(x) = f(y)$ implies $x = y$. For a function $f : X \to Y$ and a set $S \subseteq X$, $f|_S : S \to Y$ is a function such that for $s \in S$, we have $f|_S(s) = f(s)$. We let ω denote the exponent in the running time of algorithm for matrix multiplication, the current best known bound for it is $\omega < 2.373$ [35].

Graphs.

Consider a graph G. By $V(G)$ and $E(G)$ we denote the set of vertices and edges in G, respectively. For $X \subseteq V(G)$, $G[X]$ denotes the subgraph of G with vertex set X and edge set $\{uv \in E(G) \mid u, v \in X\}$. For $Y \subseteq E(G)$, $G[Y]$ denotes the subgraph of G with vertex set $\cup_{uv \in Y} \{u, v\}$ and edge set Y.

Let G be a graph. An *independent set* in G is a set $X \subseteq V(G)$ such that for every $u, v \in X$, $uv \notin E(G)$. A *matching* in G is a set $Y \subseteq E(G)$ such that no two distinct edges in Y have a common vertex. A matching M in G is a *maximum matching* if for any matching Y in G, $|M| \geq |Y|$. A matching M in G saturates a set $X \subseteq V(G)$, if every vertex in X is an end point of an edge in M. For $v_1, v_\ell \in V(G)$, a v_1v_ℓ*-path* $P = (v_1, v_2, \cdots, v_{\ell-1}, v_\ell)$ in G is a sequence of (distinct) vertices, such that $V(P) \subseteq V(G)$ and for each $i \in [\ell - 1]$, we have $v_iv_{i+1} \in E(G)$. Moreover, the edges in $\{v_iv_{i+1} \mid i \in [\ell - 1]\}$ are called edges in P. The *length* of a path is the number of edges in it. A *shortest uv-path* is a uv-path with minimum number of edges.

A *chordal graph* is a graph with no induced cycles of length at least four. An *interval graph* is an intersection graph of line segments (intervals) on the real line, i.e., its vertex set is a set of intervals, and two vertices are adjacent if and only if their corresponding intervals intersect. A *unit-interval* graph is an intersection graph of intervals of unit length on the real line. For $d \in \mathbb{N}$, a graph is *d-degenerate* if every subgraph of it has a vertex of degree at most d. A *clique* K in G is an (induced) subgraph, such that for any two distinct vertices $u, v \in V(K)$ we have $uv \in E(G)$. A vertex set $S \subseteq V(G)$ is a *clique* in G if $G[S]$ is a clique. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. If $V_1 \cap V_2 = \emptyset$, then disjoint union of G_1 and G_2 is the graph $G = (V_1 \cup V_2, E_1 \cup E_2)$. If $V_1 = V_2$, then the edge-wise union of G_1 and G_2 is the graph $G = (V_1, E_1 \cup E_2)$.

In the following we state definitions related to tree decomposition and some results on them, that are used in our algorithms.

▶ **Definition 1.** A *tree decomposition* of a graph H is a pair (T, X), where T is a rooted tree and $X = \{X_t \mid t \in V(T)\}$. Every node t of T is assigned a subset $X_t \subseteq V(H)$, called a *bag*, such that following conditions are satisfied:

- $\bigcup_{t \in V(T)} X_t = V(H)$, i.e. each vertex in H is in at least one bag;
- For every edge $uv \in E(H)$, there is $t \in V(T)$ such that $u, v \in X_t$;
- For every vertex $v \in V(H)$ the graph $T[\{t \in V(T) \mid v \in X_t\}]$ is a connected subtree of T.

To distinguish between vertices of a graph H and vertices of its tree decomposition (T, X), we refer to the vertices in T as *nodes*. Since T is a rooted tree, we have a natural parent-child and ancestor-descendant relationship among nodes in T. For a node $t \in V(T)$, by $\mathsf{desc}(t)$ we denote the set descendant of t in T (including t). For a node $t \in V(T)$ by V_t we denote the union of all bags in the subtree rooted at t i.e. $V_t = \cup_{d \in \mathsf{desc}(t)} X_d$ and by H_t we denote the graph $H[V_t]$. A *leaf* node of T is a node with degree exactly one in T, which is different from the root node. All the nodes of T which are neither the root node nor a leaf node are *non-leaf* nodes.

We now define a more structured form of tree decomposition that will be used in the algorithm.

▶ **Definition 2.** Let (T, X) be a tree decomposition of a graph H with r as the root node. Then, (T, X) is a *nice tree decomposition* if for each each leaf ℓ in T and the root r, we have that $X_\ell = X_r = \emptyset$, and each non-leaf node $t \in V(T)$ is of one of the following types:

1. **Introduce node:** t has exactly one child, say t', and $X_t = X_{t'} \cup \{v\}$, where $v \notin X_{t'}$. We say that v is *introduced* at t;
2. **Forget node:** t has exactly one child, say t', and $X_t = X_{t'} \setminus \{v\}$, where $v \in X_{t'}$. We say that v is *forgotten* at t;
3. **Join node:** t has exactly two children, say t_1 and t_2, and $X_t = X_{t_1} = X_{t_2}$.

▶ **Proposition 3** ([4, 22]). *Given a tree decomposition (T, X) of a graph H, in polynomial time we can compute a nice tree decomposition (T', X') of H that has at most $\mathcal{O}(k|V(H)|)$ nodes, where, k is the size of the largest bag in X. Moreover, for each $t' \in V(T')$, there is $t \in V(T)$ such that $X'_{t'} \subseteq X_t$.*

A tree decomposition (T, X) of a graph H, where for each $t \in V(T)$, the graph $H[X_t]$ is a clique, is called a *clique-tree*. Next, we state a result regarding computation of a clique-tree of a chordal graph.

▶ **Proposition 4** ([17]). *Given an n vertex chordal graph H, in polynomial time we can construct a clique-tree (T, X) of H with $\mathcal{O}(n)$ nodes.*

Using Proposition 3 and 4 we obtain the following result.

▶ **Proposition 5.** *Given an n vertex chordal graph H, in polynomial time we can construct a nice tree decomposition which is also a clique-tree (nice clique-tree), (T, X) of H with $\mathcal{O}(n^2)$ nodes.*

Matroids and representative sets.

In the following we state some basic definitions related to matroids. We refer the reader to [30] for more details. We also state the definition of representative families and state some results related to them.

▶ **Definition 6.** A pair $\mathcal{M} = (U, \mathcal{I})$, where U is the ground set and $\mathcal{I}$ is a family of subsets of U, is a *matroid* if the following conditions hold:

- $\emptyset \in \mathcal{I}$;
- If $I_1 \in \mathcal{I}$ and $I_2 \subseteq I_1$, then $I_2 \in \mathcal{I}$;
- If $I_1, I_2 \in \mathcal{I}$ and $|I_2| < |I_1|$, then there exists an element $x \in I_1 \setminus I_2$, such that $I_2 \cup \{x\} \in \mathcal{I}$.

An inclusion-wise maximal set in $\mathcal{I}$ is called a *basis* of $\mathcal{M}$. All bases of a matroid are of the same size. The size of a basis is called the *rank* of the matroid. For a matroid $\mathcal{M} = (U, \mathcal{I})$ and sets $P, Q \subseteq U$, we say that P *fits* Q if $P \cap Q = \emptyset$ and $P \cup Q \in \mathcal{I}$.

A matroid $\mathcal{M} = (U, \mathcal{I})$ is a *linear* (or *representable*) matroid if there is a matrix A over a field $\mathbb{F}$ with E as the set of columns, such that: 1) $|E| = |U|$; 2) there is an injective function $\varphi : U \to E$, such that $X \subseteq U$ is an independent set in $\mathcal{M}$ if and only if $\{\varphi(x) \mid x \in X\}$ is a set of linearly independent columns (over $\mathbb{F}$). In the above, we say that $\mathcal{M}$ is representable over $\mathbb{F}$, and A is one of its representation.

In the following, we define some matroids and state results regarding computation of their representations.

▶ **Definition 7** ([4, 30]). A matroid $\mathcal{M} = (U, \mathcal{I})$ is a *partition* matroid if the ground set U is partitioned into sets $U_1, U_2, \cdots, U_k$, and for each $i \in [k]$, there is an integer a_i associated with U_i. A set $S \subseteq U$ is independent in $\mathcal{M}$ if and only if for each $i \in [k]$, $|S \cap U_i| \leq a_i$.

▶ **Proposition 8** ([15, 30, 26]). *A representation of a partition matroid over $\mathbb{Q}$ (the field of rationals) can be computed in polynomial time.*

▶ **Definition 9.** Let $\mathcal{M}_1 = (U_1, \mathcal{I}_1), \mathcal{M}_2 = (U_2, \mathcal{I}_2) \cdots, \mathcal{M}_t = (U_t, \mathcal{I}_t)$ be t matroids with $U_i \cap U_j = \emptyset$, for all $1 \leq i \neq j \leq t$. The *direct sum* $\mathcal{M}_1 \oplus \cdots \oplus \mathcal{M}_t$, of $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_t$ is the matroid with ground set $U = \cup_{i \in [t]} U_i$ and $X \subseteq U$ is independent in $\mathcal{M}$ if and only if for each $i \in [t]$, $X \cap U_i \in \mathcal{I}_i$.

▶ **Proposition 10** ([26, 30]). *Given matrices $A_1, A_2, \cdots, A_t$ (over $\mathbb{F}$) representing matroids $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_t$, respectively, we can compute a representation of their direct sum, $\mathcal{M}_1 \oplus \cdots \oplus \mathcal{M}_t$, in polynomial time.*

Next, we state the definition of representative families.

▶ **Definition 11.** Let $\mathcal{M} = (U, \mathcal{I})$ be a matroid, and $\mathcal{A}$ be a p-family of U. We say that $\mathcal{A}' \subseteq \mathcal{A}$ is a q-representative for $\mathcal{A}$ if for every set $Y \subseteq U$ of size q, if there is a set $X \in \mathcal{A}$, such that $X \cap Y = \emptyset$ and $X \cup Y \in \mathcal{I}$, then there is a set $X' \in \mathcal{A}'$ such that $X' \cap Y = \emptyset$ and $X' \cup Y \in \mathcal{I}$. If $\mathcal{A}' \subseteq \mathcal{A}$ is a q-representative for $\mathcal{A}$ then we denote it by $\mathcal{A}' \subseteq^q_{rep} \mathcal{A}$.

In the following, we state some basic propositions regarding q-representative sets, which will be used later.

▶ **Proposition 12** ([4, 14]). *If $\mathcal{A}_1 \subseteq^q_{\mathsf{rep}} \mathcal{A}_2$ and $\mathcal{A}_2 \subseteq^q_{\mathsf{rep}} \mathcal{A}_3$, then $\mathcal{A}_1 \subseteq^q_{\mathsf{rep}} \mathcal{A}_3$.*

▶ **Proposition 13** ([4, 14]). *If $\mathcal{A}_1$ and $\mathcal{A}_2$ are two p-families such that $\mathcal{A}'_1 \subseteq^q_{\mathsf{rep}} \mathcal{A}_1$ and $\mathcal{A}'_2 \subseteq^q_{\mathsf{rep}} \mathcal{A}_2$, then $\mathcal{A}'_1 \cup \mathcal{A}'_2 \subseteq^q_{\mathsf{rep}} \mathcal{A}_1 \cup \mathcal{A}_2$.*

Next, we state a result regarding the computation of a q-representative set.

▶ **Theorem 14** ([4, 14]). *Given a matrix M (over field $\mathbb{F}$) representing a matroid $\mathcal{M} = (U, \mathcal{I})$ of rank k, a p-family $\mathcal{A}$ of independent sets in $\mathcal{M}$, and an integer q such that $p + q = k$, there is an algorithm which computes a q-representative family $\mathcal{A}' \subseteq^q_{\mathsf{rep}} \mathcal{A}$ of size at most $\binom{p+q}{p}$ using at most $\mathcal{O}\big(|\mathcal{A}|\big(\binom{p+q}{p}p^\omega + \binom{p+q}{p}^{\omega-1}\big)\big)$ operations over $\mathbb{F}$.*

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two families of sets over U and $\mathcal{M} = (U, \mathcal{I})$ be a matroid. We define their convolution as follows.

$$\mathcal{A}_1 \star \mathcal{A}_2 = \{A_1 \cup A_2 \mid A_1 \in \mathcal{A}_1, A_2 \in \mathcal{A}_2, A_1 \cap A_2 = \emptyset \text{ and } A_1 \cup A_2 \in \mathcal{I}\}$$

▶ **Lemma 15.** *Let $\mathcal{M} = (U, \mathcal{I})$ be a matroid, $\mathcal{A}_1$ be a p_1-family, and $\mathcal{A}_2$ be a p_2-family. If $\mathcal{A}'_1 \subseteq^{k-p_1}_{\mathsf{rep}} \mathcal{A}_1$ and $\mathcal{A}'_2 \subseteq^{k-p_2}_{\mathsf{rep}} \mathcal{A}_2$, then $\mathcal{A}'_1 \star \mathcal{A}'_2 \subseteq^{k-p_1-p_2}_{\mathsf{rep}} \mathcal{A}_1 \star \mathcal{A}_2$.*

Proof. The proof of this lemma is similar to the proof of Lemma 12.28 in [4]. Let B be a set of size $k - p_1 - p_2$. Suppose there exists a set $A_1 \cup A_2 \in \mathcal{A}_1 \star \mathcal{A}_2$ that fits B. Since, $(A_1 \cup A_2) \cap B = \emptyset$, we have $|B \cup A_2| = k - p_1$. This implies that there exists $A'_1 \in \mathcal{A}'_1$ which fits $B \cup A_2$, i.e., $(A'_1 \cup B \cup A_2) \in \mathcal{I}$ and $A'_1 \cap (B \cup A_2) = \emptyset$ which gives $|A'_1 \cup B| = k - p_2$. This means, there exists $A'_2 \in \mathcal{A}'_2$ that fits $A'_1 \cup B$, i.e., $(A'_2 \cup A'_1 \cup B) \in \mathcal{I}$ and $A'_2 \cap (A'_1 \cup B) = \emptyset$. Since, $A'_1 \cap (B \cup A_2) = \emptyset$ and $A'_2 \cap (A'_1 \cup B) = \emptyset$, we get $(A'_1 \cup A'_2) \cap B = \emptyset$. Hence, $A'_1 \cup A'_2$ fits B and $(A'_1 \cup A'_2) \in \mathcal{A}'_1 \star \mathcal{A}'_2$. ◀

Next, we give a result regarding computation of convolution ($\star$).

▶ **Proposition 16.** *Let M be a matrix over a field $\mathbb{F}$ representing a matroid $\mathcal{M} = (U, \mathcal{I})$ over an n-element ground set, $\mathcal{A}_1$ be a p_1-family, and $\mathcal{A}_2$ be a p_2-family, where $p_1 + p_2 = k$. Then $\mathcal{A}_1 \star \mathcal{A}_2$ can be computed in time $\mathcal{O}(2^k n^{\mathcal{O}(1)})$.*

Proof. Consider the standard convolution operation, $\mathcal{A}_1 \circ \mathcal{A}_2 = \{A_1 \cup A_2 \mid A_1 \in \mathcal{A}_1, A_2 \in \mathcal{A}_2$ and $A_1 \cap A_2 = \emptyset\}$ defined in [4, Section 12.3.5]. The family $\mathcal{A}_1 \circ \mathcal{A}_2$ can be computed in $\mathcal{O}(2^k n^3)$ time [4, Exercise 12.12]. Since, $\mathcal{A}_1 \star \mathcal{A}_2 = \{A_1 \cup A_2 \mid A_1 \in \mathcal{A}_1, A_2 \in \mathcal{A}_2, A_1 \cap A_2 = \emptyset$, and $A_1 \cup A_2 \in \mathcal{I}\}$. Hence, $X \in \mathcal{A}_1 \star \mathcal{A}_2$ if and only if $X \in \mathcal{A}_1 \circ \mathcal{A}_2$ and X is a set of linearly independent columns (over $\mathbb{F}$). Testing whether a set of vectors is linearly independent over a field can be done in time polynomial in size of the set (using Gaussian elimination). Therefore, testing if an $X \in \mathcal{A}_1 \circ \mathcal{A}_2$ is linearly independent, can be done in time $\mathcal{O}(n^{\mathcal{O}(1)})$. Since $|\mathcal{A}_1 \circ \mathcal{A}_2| \leq |\mathcal{A}_1||\mathcal{A}_2|$, family $\mathcal{A}_1 \star \mathcal{A}_2$ can be computed in $\mathcal{O}((2^k + |\mathcal{A}_1||\mathcal{A}_2|)n^{\mathcal{O}(1)})$ time. Since, $|\mathcal{A}_1| \leq 2^{p_1}$ and $|\mathcal{A}_2| \leq 2^{p_2}$, the running time is bounded by $\mathcal{O}(2^k n^{\mathcal{O}(1)})$. ◀

Universal sets and their computation.

▶ **Definition 17.** An (n, k)-*universal set* is a family $\mathcal{F}$ of subsets of $[n]$ such that for any set $S \subseteq [n]$ of size k, the family $\{A \cap S \mid A \in \mathcal{F}\}$ contains all 2^k subsets of S.

Next, we state a result regarding the computation of a universal set.

▶ **Proposition 18** ([4, 28])**.** *For any $n, k \geq 1$, we can compute an (n, k)-universal set of size $2^k k^{\mathcal{O}(\log k)} \log n$ in time $2^k k^{\mathcal{O}(\log k)} n \log n$.*

3 W[1]-hardness Results

In this section, we show that Conflict Free Maximum Matching and Conflict Free Shortest Path are W[1]-hard, when parameterized by the solution size.

3.1 W[1]-hardness of CF-MM

We show that CF-MM is W[1]-hard, when parameterized by the solution size, even when the graph where we want to find a matching, is itself a matching (disjoint union of edges). To prove our result, we give an appropriate reduction from Independent Set to CF-MM. In the following, we define the problem Independent Set.

> Independent Set **Parameter:** k
> **Input:** A graph G and an integer k.
> **Question:** Is there a set $X \subseteq V(G)$ of size at least k such that X is an independent set in G?

It is known that Independent Set is W[1]-hard, when parameterized by the size of an independent set [4, 8].

▶ **Theorem 19.** CF-MM *is* W[1]*-hard, when parameterized by the solution size.*

Proof. Given an instance $(G^\star, k)$ of Independent Set, we construct an equivalent instance (G, H, k) of CF-MM as follows. We first describe the construction of G. For each $v \in V(G^\star)$, we add an edge vv' to G. Notice that G is a matching. This completes the description of G. Next, we move to the construction of H. We have $V(H) = \{e_v = vv' \mid v \in V(G^\star)\}$. Moreover, for $e_u, e_v \in V(H)$, we add the edge $e_u e_v$ to $E(H)$ if and only if $uv \in E(G^\star)$. We note that H is exactly the same as $G^\star$, with vertices being renamed. This completes the construction of (G, H, k) of CF-MM. Next, we show that $(G^\star, k)$ is a yes instance of Independent Set if and only if (G, H, k) is a yes instance of CF-MM.

In forward direction, let $(G^\star, k)$ be a yes instance of Independent Set, and S be one of its solution. Let $S' = \{e_v \mid v \in S\}$. We show that S' is a solution to CF-MM. Notice that by construction, S' is a matching in G, and $|S'| = |S| \geq k$. Moreover, $G^\star$ is isomorphic to H, with the vertex mapping as $\varphi : V(G^\star) \to V(H)$, where for $v \in V(G^\star)$, $\varphi(v) = e_v$. Hence, S' is an independent set in H.

In reverse direction, let (G, H, k) be a yes instance of CF-MM, and S' be one of its solution. Let $S = \{v \mid e_v \in S'\}$. Using an analogous argument as in the forward direction, we conclude that S is a solution to Independent Set in $(G^\star, k)$. This concludes the proof. ◀

3.2 W[1]-hardness of CF-SP

We show that CF-SP is W[1]-hard, when parameterized by the solution size, even when the conflict graph is a proper interval graph. We refer to this restricted variant of the problem as Unit Interval CF-SP. To prove our result, we give an appropriate reduction from a multicolored variant of the problem Unit 2-Track Independent Set, which we call Unit 2-Track Multicolored IS. In the following, we define the problems Unit 2-Track Independent Set and Unit 2-Track Multicolored IS.

Unit 2-Track Independent Set (Unit 2-Track IS) **Parameter:** k
Input: Two unit-interval graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, and an integer k.
Question: Is there a set $S \subseteq V$ of size at least k, such that S is an independent set in both G_1 and G_2?

Unit 2-Track Multicolored IS (Unit 2-Track MIS) **Parameter:** k
Input: Two unit-interval graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, and a partition $V_1, V_2, \cdots, V_k$ of V.
Question: Is there a set $S \subseteq V$, such that S is an independent set in both G_1 and G_2, and for each $i \in [k]$, we have $|S \cap V_i| = 1$?

It is known that Unit 2-Track IS is W[1]-hard, when parameterized by the solution size [21]. We show that the problem Unit 2-Track MIS is W[1]-hard, when parameterized by the number of sets in the partition. We show this by giving an appropriate (Turing) reduction from Unit 2-Track IS. Finally, we give a reduction from Unit 2-Track MIS to Unit Interval CF-SP, hence obtaining the desired result.

3.3 W[1]-hardness of Unit 2-Track MIS.

We give a (Turing) reduction from Unit 2-Track IS to Unit 2-Track MIS. Moreover, since we want to rule out existence of an FPT algorithm, we spend FPT time to obtain FPT many instances of Unit 2-Track MIS.

Before proceeding to the reduction from Unit 2-Track IS to Unit 2-Track MIS, we define the notion of *perfect hash family*, which will be used in the reduction.

▶ **Definition 20.** An (n, k)-*perfect hash family* $\mathcal{F}$, is a family of functions $f : [n] \to [k]$ such that for every set $S \subseteq [n]$ of size k, there is an $f \in \mathcal{F}$, such that $f|_S$ is injective.

In the following, we state a result regarding computation of an (n, k)-perfect hash family.

▶ **Theorem 21.** [4, 28] *For any $n, k \geq 1$, an (n, k)-perfect hash family of size $e^k k^{\mathcal{O}(\log k)} \log n$ can be constructed in $e^k k^{\mathcal{O}(\log k)} n \log n$ time.*

Now we are ready to give a (Turing) reduction from Unit 2-Track IS to Unit 2-Track MIS.

▶ **Lemma 22.** *There is a parameterized Turing reduction from* Unit 2-Track IS *to* Unit 2-Track MIS.

Proof. Let (G_1, G_2, k) be an instance of Unit 2-Track IS, where $V(G_1) = V(G_2) = [n]$. We construct a family $\mathcal{C}$ of instances of Unit 2-Track MIS as follows. We start by computing an (n, k)-perfect hash family $\mathcal{F}$, of size $e^k k^{\mathcal{O}(\log k)} \log n$, in time $e^k k^{\mathcal{O}(\log k)} n \log n$, using Theorem 21. Now, for each $f \in \mathcal{F}$, we construct an instance $I_f = (G_1, G_2, V_1^f, V_2^f, \cdots, V_k^f)$ of Unit 2-Track MIS as follows. For $i \in [k]$, we set $V_i^f = \{v \in V(G_1) \mid f(v) = i\}$. Finally, we set $\mathcal{C} = \{I_f \mid f \in \mathcal{F}\}$.

We claim that (G_1, G_2, k) is a yes instance of Unit 2-Track IS if and only if there is $I_f \in \mathcal{C}$ such that I_f is a yes instance of Unit 2-Track MIS.

In the forward direction, let (G_1, G_2, k) be a yes instance of Unit 2-Track IS, and S be one of its solution of size k. Consider $f \in \mathcal{F}$ such that $f|_S$ is injective, which exists since $\mathcal{F}$ is an (n, k)-perfect hash family. By construction of $\mathcal{C}$, we have $I_f \in \mathcal{C}$. Moreover, by construction of f, for each $i \in [k]$, we have $|S \cap V_i| = 1$. Hence, S is a solution to I_f.

In the reverse direction, let $I_f \in \mathcal{C}$ be a yes instance of Unit 2-Track MIS, and S be one of its solution. Clearly, S is a solution to Unit 2-Track IS in (G_1, G_2, k) as $I_f = (G_1, G_2, V_1^f, V_2^f, \cdots, V_k^f)$. This concludes the proof. ◀

▶ **Theorem 23.** Unit 2-Track MIS *is* W[1]-hard*, when parameterized by the solution size.*

Proof. Follows from Lemma 22 and W[1]-hardness of Unit 2-Track IS. ◀

3.4 W[1]-hardness of Unit Interval CF-SP

We give a parameterized reduction from Unit 2-Track MIS to Unit Interval CF-SP. Let $(G_1, G_2, V_1, \cdots, V_k)$ be an instance of Unit 2-Track MIS. We construct an instance (G', H, s, t, k') of Unit Interval CF-SP as follows. For each $v \in V(G_1)$, we add a path on 3 vertices namely, (v_1, v_2, v_3) in G'. For notational convenience, for $v \in V(G_1)$, by $e_{12}(v)$ and $e_{23}(v)$ we denote the edges v_1v_2 and v_2v_3, respectively. Consider $i \in [k-1]$. For $u \in V_i$ and $v \in V_{i+1}$, we add the edge $z_{uv} = u_3v_1$ to $E(G')$ (see Figure 1). Moreover, by Z_i, we denote the set $\{z_{uv} \mid u \in V_i, v \in V_{i+1}\}$. We add two new vertices s and t to $V(G')$, and add all the edges in $Z_0 = \{sv_1 \mid v \in V_1\}$ and $Z_k = \{v_3t \mid v \in V_k\}$ to $E(G')$. Next, we move to the construction of H. Note that H must be a unit-interval graph on the vertex set $E(G') = (\cup_{i\in[0,k]} Z_i) \cup (\cup_{v\in V(G_1)} \{e_{12}(v), e_{23}(v)\})$. In H, each vertex in $\cup_{i\in[0,k]} Z_i$ is an isolated vertex. Let $E_{12} = \{e_{12}(v) \mid v \in V(G_1)\}$ and $E_{23} = \{e_{23}(v) \mid v \in V(G_1)\}$. For $e_{12}(u), e_{12}(v) \in E_{12}$, we add the edge $e_{12}(u)e_{12}(v)$ to $E(H)$ if and only if $uv \in E(G_1)$. Similarly, for $e_{23}(u), e_{23}(v) \in E_{23}$, we add the edge $e_{23}(u)e_{23}(v)$ to $E(H)$ if and only if $uv \in E(G_2)$. Observe that $H[E_{12}]$ is isomorphic to G_1, with bijection $\phi_1 : V(G_1) \to E_{12}$ with $\phi_1(v) = e_{12}(v)$. Similarly, $H[E_{23}]$ is isomorphic to G_2 with bijection $\phi_2 : V(G_2) \to E_{23}$ with $\phi_2(v) = e_{23}(v)$. By construction, H is disjoint union of unit-interval graphs, and hence is a unit-interval graph. Finally, we set $k' = 3k + 1$. This completes the description of the reduction.

In the following lemma we show that the instance $(G_1, G_2, V_1, \cdots, V_k)$ of Unit 2-Track MIS and the instance (G', H, s, t, k') of Unit Interval CF-SP are equivalent.

▶ **Lemma 24.** $(G_1, G_2, V_1, \cdots, V_k)$ *is a* yes *instance of* Unit 2-Track MIS *if and only if* (G', H, s, t, k') *is a* yes *instance of* Unit Interval CF-SP.

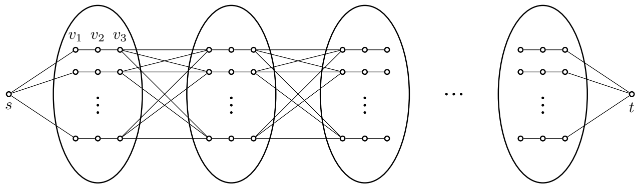

Figure 1 An illustration of the construction of G' in W[1]-hardness of Unit Interval CF-SP.

Proof. In the forward direction, let $(G_1, G_2, V_1, \cdots, V_k)$ be a yes instance of Unit 2-Track MIS, and $S = \{v^1, v^2, \cdots, v^k\}$ be one of its solution, such that $v^i \in V_i$. We claim that $P = (s, v_1^1, v_2^1, v_3^1, \cdots, v_1^k, v_2^k, v_3^k, t)$ is a conflict-free path (on $3k + 1$ edges) in G'. By the construction of G', it follows that P is a path in G'. Next, we show that $E(P)$ is an independent set in H. Let $v_3^0 = s$ and $v_1^{k+1} = t$. By construction, each edge in $\{v_3^i v_1^{i+1} \mid i \in [0, k]\} \subseteq \cup_{[0,k]} Z_i$ is an isolated vertex in H. Also, for each $i \in [k]$, we have that $\{e_{12}(v^i), e_{23}(v^i)\}$ is an independent set in H. Next, consider $i, j \in [k]$, where $i \neq j$. By construction $e_{12}(v^i)e_{23}(v^j), e_{23}(v^i)e_{12}(v^j) \notin E(H)$. Moreover, $e_{12}(v^i)e_{12}(v^j) \notin E(H)$ since S in an independent set in G_1. Similarly, $e_{23}(v^i)e_{23}(v^j) \notin E(H)$ as S is an independent set in G_2. In the above, we have considered every pair of edges in $E(P)$, and argued that no two of them are adjacent to each other in H. Hence, it follows that P is a solution to Unit Interval CF-SP in (G', H, s, t, k').

In the reverse direction, let P be a solution to Unit Interval CF-SP in (G', H, s, t, k'). By the construction of G', the path P must be of the form $(s, v_1^1, v_2^1, v_3^1, \cdots, v_1^k, v_2^k, v_3^k, t)$. We claim that $S = \{v^1, v^2, \cdots, v^k\}$ is an independent set in both G_1 and G_2. Suppose not, then there is an edge $v^i v^j$, $i \neq j$ and $i, j \in [k]$ say, in G_1 (the case when it is in G_2 is symmetric). But then $e_{12}(v^i)e_{12}(v^j)$ is an edge in H, contradicting that $E(P)$ is an independent set in H. Hence, we have that S is an independent set both in G_1 and G_2. Moreover, since P is a path of length at most $3k + 1$, it must hold that for each $i \in [k]$, we have $v^i \in V_i$. Hence, S is a solution to Unit 2-Track MIS in $(G_1, G_2, V_1, \cdots, V_k)$. ◀

▶ **Theorem 25.** Unit Interval CF-SP *is* W[1]-hard*, when parameterized by the solution size.*

Proof. Follows from the construction of instance (G', H, s, t, k') of Unit Interval CF-SP, for the given instance $(G_1, G_2, V_1, \cdots, V_k)$ of Unit 2-Track MIS, Lemma 24, and Theorem 23. ◀

4 FPT Algorithm for CF-MM with Chordal Conflict

In this section, we show that CF-MM is FPT, when the conflict graph belongs to the family of chordal graphs. We call this restricted version of CF-MM as Chordal Conflict Matching. Towards designing an algorithm for Chordal Conflict Matching, we first give an FPT algorithm for a restricted version of Chordal Conflict Matching, where we want to compute a matching for a bipartite graph. We call this variant of Chordal Conflict Matching as Chordal Conflict Bipartite Matching (CCBM). We then employ the algorithm for CCBM to design an FPT algorithm for Chordal Conflict Matching.

4.1 FPT algorithm for CCBM

We design an FPT algorithm for the problem CCBM, where the conflict graph is chordal and the graph where we want to compute a matching is a bipartite graph. The problem CCBM is formally defined below.

> Chordal Conflict Bipartite Matching (CCBM) **Parameter:** k
> **Input:** A bipartite graph $G = (V, E)$ with vertex bipartition L, R, a conflict graph $H = (E, E')$, and an integer k.
> **Question:** Is there a matching $M \subseteq E$ of size k in G, such that M is an independent set in H?

The FPT algorithm for CCBM is based on a dynamic programming routine over a tree decomposition of the conflict graph H and representative sets on the graph G. Let (G, L, R, H, k) be an instance of CF-MM, where G is a bipartite graph on n vertices, with vertex bipartition L, R, and H is a chordal graph with $V(H) = E(G)$.

In the following, we construct three matroids $\mathcal{M}_L = (E, \mathcal{I}_L), \mathcal{M}_R = (E^c, \mathcal{I}_R)$, and $\mathcal{M} = (E \cup E^c, \mathcal{I})$. Matroids $\mathcal{M}_L$ and $\mathcal{M}_R$ are partition matroids and the matroid $\mathcal{M}$ is the direct sum of $\mathcal{M}_L$ and $\mathcal{M}_R$. The ground set of $\mathcal{M}_L$ is $E = E(G)$. The set E^c contains a copy of edges in E, i.e., $E^c = \{e^c \mid e \in E\}$. We create two (disjoint) sets E and E^c, because $\mathcal{M}$ is the direct sum of $\mathcal{M}_L$ and $\mathcal{M}_R$, and we want their ground sets to be disjoint. Next, we describe the partition $\mathcal{E}$ of E into $|L|$ sets and $|L|$ integers, one for each set in the partition, for the partition matroid $\mathcal{M}_L$. For $u \in L$, let $E_u = \{uv \mid uv \in E\}$. Notice that for $u, v \in L$, where $u \neq v$, we have $E_u \cap E_v = \emptyset$. Moreover, $\cup_{u \in E} E_u = E$. We let $\mathcal{E} = \{E_u \mid u \in L\}$, and for each $u \in L$, we set $a_u = 1$. Similarly, we define the partition $\mathcal{E}^c$ of E^c with respect to set R. That is, we let $\mathcal{E}^c = \{E^c_u = \{(uv)^c \mid uv \in E(G)\} \mid u \in R\}$. Furthermore, for $u \in R$, we let $a_{u^c} = 1$. We define the following notation, which will be used later. For $Z \subseteq E$, we let $Z^c = \{e^c \mid e \in Z\} \subseteq E^c$.

▶ **Proposition 26.** *$Q \subseteq E(G)$ is a matching in G with vertex bipartition L and R if and only if $Q \cup Q^c$ is an independent set in the matroid $\mathcal{M} = \mathcal{M}_L \oplus \mathcal{M}_R$.*

Proof. In the forward direction, let Q be a matching in the bipartite graph $G = (V, E)$, where $V = L \cup R$. Since, $\mathcal{M}_L = (E, \mathcal{I}_L)$ is a partition matroid with partition $\mathcal{E} = \{E_u \mid u \in L\}$ and $a_u = 1$, for each $u \in L$, $Q \cap L$ is an independent set in $\mathcal{M}_L$. Similarly, $Q^c \cap R$ is an independent in $\mathcal{M}_R$. Since, $\mathcal{M} = \mathcal{M}_L \oplus \mathcal{M}_R$, it follows that $Q \cup Q^c$ is an independent set in $\mathcal{M}$.

In the reverse direction, consider $Q \subseteq E$ such that $Q \cup Q^c$ is an independent set in $\mathcal{M}$. Since, $\mathcal{M} = \mathcal{M}_\mathcal{L} \oplus \mathcal{M}_\mathcal{R}$, Q is independent in $\mathcal{M}_L$ and Q^c is independent in $\mathcal{M}_R$. Since, Q and Q^c both have copies of the same edge, no two edges in Q share an end point in G. Hence, Q forms a matching in G. ◀

To capture the independence property on the conflict graph, we rely on the fact that a chordal graph admits a nice clique-tree (Proposition 5). This allows us to do dynamic programming over a nice clique-tree. At each step of our dynamic programming routine, using representative sets, we ensure that we store a family of sets which are enough to recover (some) independent set in $\mathcal{M}$, if a solution exists.

We now move to the formal description of the algorithm. The algorithm starts by computing a nice clique-tree (T, X) of H in polynomial time, using Proposition 5. Let $r \in V(T)$ be the root of the (rooted) tree T. For $X_t \in X$, we let $\mathcal{X}_t = \{\emptyset\} \cup \{\{v\} \mid v \in X_t\}$. Recall that for $t \in V(T)$, H_t is the graph $H[V_t]$, where $V_t = \cup_{d \in \mathsf{desc}(t)} X_d$.

In the following, we state some notations, which will be used in the algorithm. For each $t \in V(T)$, $Y \in \mathcal{X}_t$, and an integer $p \in [0,k]$ we define a family $\mathcal{P}^p_{t,Y}$ as follows.

$$\mathcal{P}^p_{t,Y} = \{Z \cup Z^c \mid Z \subseteq V(H_t)(\subseteq E), |Z| = p, Z \cap X_t = Y, Z \cup Z_c \in \mathcal{I} \text{ and } H_t[Z] \text{ is edgeless}\}$$

For a family $\mathcal{F}$ of subsets of $E \cup E^c$, $\mathcal{F}$ is called a *paired-family* if for each $F \in \mathcal{F}$, there is $Z \subseteq E$, such that $F = Z \cup Z^c$.

In the following definition, we state the entries in our dynamic programming routine.

▶ **Definition 27.** *For each $t \in V(T)$, $Y \in \mathcal{X}_t$ and $p \in [0,k]$, we have an entry $c[t,Y,p]$, which stores a paired-family $\mathcal{F}(t,Y,p)$ of subsets of $E \cup E^c$ of size $2p$, such that for each $F = Z \cup Z^c \in \mathcal{F}$, the following conditions are satisfied.*

1. *$|Z| = p$;*
2. *$Z \cap X_t = Y$;*
3. *Z is a matching in G, i.e., Z and Z^c are independent sets in $\mathcal{M}_L$ and $\mathcal{M}_R$, respectively;*
4. *Z is an independent set in H_t.*

Moreover, $\mathcal{F} \neq \emptyset$ if and only if $\mathcal{P}^p_{t,Y} \neq \emptyset$.

Consider $t \in V(T)$, $Y \in \mathcal{X}_t$ and $p \in [0,k]$. Observe that $\mathcal{P}^p_{t,Y}$ is a valid candidate for $c[t,Y,p]$, which also implies that (G,H,k) is a yes instance of CCBM if and only if $c[r,\emptyset,k] \neq \emptyset$. However, we cannot set $c[t,Y,p] = \mathcal{P}^p_{t,Y}$ as the size of $\mathcal{P}^p_{t,Y}$ could be exponential in n, and the goal here is to obtain an FPT algorithm. Hence, we will store a much smaller subfamily (of size at most $\binom{2k}{2p}$) of $\mathcal{P}^p_{t,Y}$ in $c[t,Y,p]$, which will be computed using representative sets. Moreover, as we have a structured form of a tree decomposition (nice clique-tree) of H, we compute the entries of the table based on the entries of its children, which will be given by recursive formulae. For leaf nodes, which form base cases for recursive formulae, we compute all entries directly.

Next, we give (recursive) formulae for the computation of the table entries. Consider $t \in V(T)$, $Y \in \mathcal{X}_t$ and $p \in [0,k]$. We compute the entry $c[t,Y,k]$ based on the following cases.

Leaf node: t is a leaf node. In this case, we have $X_t = \emptyset$, and hence $\mathcal{X}_t = \{\emptyset\}$. If $p = 0$, then $\mathcal{P}^p_{t,\emptyset} = \{\emptyset\}$, and $\mathcal{P}^p_{t,\emptyset} = \emptyset$, otherwise. Since, $\mathcal{P}^p_{t,\emptyset}$ is a valid candidate for $c[t,Y,p]$, we set $c[t,Y,p] = \mathcal{P}^p_{t,\emptyset}$. Note that $c[t,Y,p]$ has size at most $1 \leq \binom{2k}{2p}$, and we can compute $c[t,Y,p]$ in polynomial time.

Introduce node: Suppose t is an introduce node with child t' such that $X_t = X_{t'} \cup \{e\}$, where $e \notin X_{t'}$. If $Y \neq \emptyset$ and $p < 1$, then we set $c[t,Y,p] = \emptyset$. Otherwise, we compute the entry as described below. Before computing the entry $c[t,Y,p]$, we first compute a set $\widetilde{\mathcal{P}}^p_{t,Y}$ as follows.

$$\widetilde{\mathcal{P}}^p_{t,Y} = \begin{cases} c[t',Y,p] & \text{if } Y \neq \{e\}; \\ c[t',\emptyset,p-1] \star \{\{e,e^c\}\} & \text{otherwise.} \end{cases} \quad (1)$$

Next, we compute $\widehat{\mathcal{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} \widetilde{\mathcal{P}}^p_{t,Y}$ of size $\binom{2k}{2p}$, using Theorem 14. Finally, we set $c[t,Y,p] = \widehat{\mathcal{P}}^p_{t,Y}$.

Correctness: To show the correctness, it is enough to show that $c[t,Y,p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$. If $Y \neq \emptyset$ and $p < 1$, then we correctly set $c[t,Y,p] = \emptyset$. Hereafter, we assume that $Y \neq \emptyset$

then $p \geq 1$. If $Y \neq \{e\}$, then the claim follows from the fact that $c[t, Y, p] = c[t', Y, p]$ and $\mathcal{P}^p_{t,Y} = \mathcal{P}^p_{t',Y}$. Therefore, we consider the case when $Y = \{e\}$. In this case, we observe the following towards proving the claim.

1. $\mathcal{P}^p_{t,Y} = \mathcal{P}^{p-1}_{t',\emptyset} \star \{\{e, e^c\}\}$.
2. $c[t', \emptyset, p-1] \subseteq^{2k-2(p-1)}_{\mathsf{rep}} \mathcal{P}^{p-1}_{t',\emptyset}$.

From item 1 and 2, and Lemma 15, it follows that $c[t', \emptyset, p-1] \star \{\{e, e^c\}\} \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$. This together with Proposition 12, and the fact that $\widehat{\mathcal{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} c[t', \emptyset, p-1] \star \{\{e, e^c\}\}$ implies that $c[t, Y, p] = \widehat{\mathcal{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$.

Forget node: Suppose t is a forget node with child t' such that $X_t = X_{t'} \setminus \{e\}$, where $e \in X_{t'}$. Before computing the entry $c[t, Y, p]$, we first compute a set $\widetilde{\mathcal{P}}^p_{t,Y}$ as follows.

$$\widetilde{\mathcal{P}}^p_{t,Y} = \begin{cases} c[t', Y, p] & \text{if } Y \neq \emptyset; \\ c[t', \emptyset, p] \cup c[t', \{e\}, p] & \text{otherwise.} \end{cases} \tag{2}$$

Next, we compute $\widehat{\mathcal{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} \widetilde{\mathcal{P}}^p_{t,Y}$ of size $\binom{2k}{2p}$, using Theorem 14. Finally, we set $c[t, Y, p] = \widehat{\mathcal{P}}^p_{t,Y}$.

Correctness: To show the correctness, it is enough to show that $c[t, Y, p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$. If $Y \neq \emptyset$, then the claim follows from the fact that $c[t, Y, p] = c[t', Y, p]$, and $\mathcal{P}^p_{t,Y} = \mathcal{P}^p_{t',Y}$. Therefore, we consider the case when $Y = \emptyset$. In this case, we observe the following towards proving the claim.

1. $c[t', \emptyset, p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t',\emptyset}$.
2. $c[t', \{e\}, p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t',\{e\}}$.
3. $\mathcal{P}^p_{t,Y} = \mathcal{P}^p_{t',\emptyset} \cup \mathcal{P}^p_{t',\{e\}}$.

From item 1 to 3, and Proposition 13, it follows that $c[t', \emptyset, p] \cup c[t', \{e\}, p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$. This together with Proposition 12, and the fact that $\widehat{\mathcal{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} c[t', \emptyset, p] \cup c[t', \{e\}, p]$ implies that $c[t, Y, p] = \widehat{\mathcal{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$.

Join node: Suppose t is a join node with children t_1 and t_2, such that $X_t = X_{t_1} = X_{t_2}$. If $Y \neq \emptyset$ and $p < 1$, then we set $c[t, Y, p] = \emptyset$. Otherwise, we compute the entry as described below. Before computing the entry $c[t, Y, p]$, we first compute a set $\widetilde{\mathcal{P}}^p_{t,Y}$ as follows.

$$\tilde{\mathcal{P}}^p_{t,Y} = \begin{cases} \bigcup\limits_{i \in [0,p]} (c[t_1, \emptyset, i] \star c[t_2, \emptyset, p-i]) & \text{if } Y = \emptyset; \\ \bigcup\limits_{i \in [p]} (c[t_1, Y, i] \star c[t_2, \emptyset, p-i]) & \text{otherwise.} \end{cases} \tag{3}$$

Next, we compute $\widehat{\mathcal{P}}^p_{t,Y} \subseteq^{2k-2p}_{\mathsf{rep}} \widetilde{\mathcal{P}}^p_{t,Y}$ of size $\binom{2k}{2p}$, using Theorem 14. Finally, we set $c[t, Y, p] = \widehat{\mathcal{P}}^p_{t,Y}$.

Correctness: To show the correctness, it is enough to show that $c[t, Y, p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$. If $Y \neq \emptyset$ and $p < 1$, then we correctly set $c[t, Y, p] = \emptyset$. Hereafter, we assume that whenever $Y \neq \emptyset$, we have $p \geq 1$. Next, we consider the following cases depending on whether or not $Y = \emptyset$.

- $Y = \emptyset$. In this case, we have $\mathcal{P}^p_{t,Y} = \cup_{i\in[0,p]}(\mathcal{P}^i_{t_1,\emptyset} \star \mathcal{P}^{p-i}_{t_1,\emptyset})$. Moreover, for $i \in [0,p]$, we have that $c[t_1,\emptyset,i] \subseteq^{2k-2i}_{\mathsf{rep}} \mathcal{P}^i_{t_1,\emptyset}$ and $c[t_2,\emptyset,p-i] \subseteq^{2k-2(p-i)}_{\mathsf{rep}} \mathcal{P}^{p-i}_{t_1,\emptyset}$. Above arguments together with Proposition 13 and Lemma 15 implies that $c[t,Y,p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$.
- $Y \neq \emptyset$. In this case, we start by arguing that $\widehat{\mathcal{P}}^p_{t,Y} = \cup_{i\in[p]}(c[t_1,Y,i] \star c[t_2,\emptyset,p-i]) \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$. To this end, consider a set $A \in \mathcal{P}^p_{t,Y}$ of size $2p$ and a set $B \subseteq E \cup E^c$ of size $2k-2p$ such that $A \cup B \in \mathcal{I}$ and $A \cap B = \emptyset$. Observe that by construction of $\mathcal{P}^p_{t,Y}$, $A \subseteq V(H_t) \cup (V(H_t))^c$, $A \cap X_t = Y$. Let $A_1 = A \cap V(H_{t_1})$, $A_2 = A \setminus A_1$, and $i^* = |A_1|$. Since $A \in \mathcal{P}^p_{t,Y}$, and $\mathcal{P}^p_{t,Y}$ is a paired-family, it holds that $A^c_1 \cup A^c_2 \subseteq A$. Let $B_2 = B \cup A_1 \cup A^c_1$, and note that $|B_2| = 2k - 2(p-i^*)$. Moreover, $c[t_2,\emptyset,p-i^*] \subseteq^{2k-2(p-i^*)}_{\mathsf{rep}} \mathcal{P}^{p-i^*}_{t_2,\emptyset}$, and therefore, there is $\tilde{A}_2 \cup \tilde{A}^c_2 \in c[t_2,\emptyset,p-i^*]$ such that $(\tilde{A}_2 \cup \tilde{A}^c_2) \cup B_2 \in \mathcal{I}$ and $(\tilde{A}_2 \cup \tilde{A}^c_2) \cap B_2 = \emptyset$. Next, consider $B_1 = B \cup (\tilde{A}_2 \cup \tilde{A}^c_2)$, and note that $|B_1| = 2k - 2p + 2(p-i^*) = 2k - 2i^*$. Since, $c[t_1,Y,i^*] \subseteq^{2k-2i^*}_{\mathsf{rep}} \mathcal{P}^{i^*}_{t_1,Y}$, therefore, there is $\tilde{A}_1 \cup \tilde{A}^c_1 \in c[t_1,T,i^*]$ such that $B_1 \cup (\tilde{A}_1 \cup \tilde{A}^c_1) \in \mathcal{I}$ and $B_1 \cap (\tilde{A}_1 \cup \tilde{A}^c_1) = \emptyset$. The above arguments imply that $(\tilde{A}_1 \cup \tilde{A}^c_1) \cup (\tilde{A}_2 \cup \tilde{A}^c_2) \in \mathcal{I}$ and $(\tilde{A}_1 \cup \tilde{A}^c_1) \cap (\tilde{A}_2 \cup \tilde{A}^c_2) = \emptyset$. Hence, by definition of the convolution operation ($\star$), we have $(\tilde{A}_1 \cup \tilde{A}^c_1) \cup (\tilde{A}_2 \cup \tilde{A}^c_2) \in c[t_1,Y,i^*] \star c[t_2,\emptyset,p-i^*]$. Moreover, $B \cup (\tilde{A}_1 \cup \tilde{A}^c_1) \cup (\tilde{A}_2 \cup \tilde{A}^c_2) \in \mathcal{I}$ and $B \cap (\tilde{A}_1 \cup \tilde{A}^c_1) \cup (\tilde{A}_2 \cup \tilde{A}^c_2) = \emptyset$. Therefore, $\cup_{i\in[p]}(c[t_1,Y,i] \star c[t_2,\emptyset,p-i]) \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$. This together with Proposition 12 implies that $c[t,Y,p] \subseteq^{2k-2p}_{\mathsf{rep}} \mathcal{P}^p_{t,Y}$.

This completes the description of the (recursive) formulae and their correctness for computing all entries of the table. The correctness of the algorithm follows from the correctness of the (recursive) formulae, and the fact that (G,H,k) is a yes instance of CCBM if and only if $c[r,\emptyset,k] \neq \emptyset$. Next, we move to the running time analysis of the algorithm.

▶ **Lemma 28.** *The algorithm presented for* CCBM *runs in time $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$, where n is the number of vertices in G.*

Proof. We do the running time analysis based on time required to compute an entry $c[t,Y,k]$, for $t \in V(T)$, $Y \in \mathcal{X}_t$ and $p \in [0,k]$. We consider the following cases.

Leaf node: For leaf nodes the entries $c[t,Y,k]$ can be computed in polynomial time.

Introduce node: The algorithm first computes a family $\widetilde{\mathcal{P}}^p_{Y,t}$ from Equation 1, which takes $2^{2k} n^{\mathcal{O}(1)}$ time (using Proposition 16). Moreover, $|\widetilde{\mathcal{P}}^p_{Y,t}| \leq \max\{\binom{2k}{2p}, \binom{2k}{2(p-1)}\}$. The algorithm then computes $\hat{\mathcal{P}}^p_{Y,t} \subseteq^{2k-2p}_{rep} \widetilde{\mathcal{P}}^p_{Y,t}$ using Theorem 14, which takes time bounded by $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$.

Forget node: The algorithm first computes a family $\widetilde{\mathcal{P}}^p_{Y,t}$ from Equation 1, which takes at most $\binom{2k}{2p}$ time by standard set union operation. Moreover, $|\widetilde{\mathcal{P}}^p_{Y,t}| \leq 2\binom{2k}{2p}$. The algorithm then computes $\hat{\mathcal{P}}^p_{Y,t} \subseteq^{2k-2p}_{rep} \widetilde{\mathcal{P}}^p_{Y,t}$. This takes time $|\widetilde{\mathcal{P}}^p_{Y,t}|\binom{2k}{2p}^{\omega-1} n^{\mathcal{O}(1)} \leq \binom{2k}{2p}^{\omega} n^{\mathcal{O}(1)} \leq 4^{\omega k} n^{\mathcal{O}(1)}$. Therefore, the time required to compute an entry at forget node is at most $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$.

Join node: The algorithm first computes a family $\widetilde{\mathcal{P}}^p_{Y,t}$ from Equation 3, which takes at most $2^{2k} n^{\mathcal{O}(1)}$ time by Proposition 16 and standard set union operation. Moreover, $|\widetilde{\mathcal{P}}^p_{Y,t}| \leq 2^{\mathcal{O}(\omega k)}$. Now the algorithm applies Theorem 14 on $\tilde{\mathcal{P}}^p_{Y,t}$ and computes $\hat{\mathcal{P}}^p_{Y,t} \subseteq^{2k-2p}_{rep} \tilde{\mathcal{P}}^p_{Y,t}$. This takes time bounded by $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$. Therefore, the time required to compute an entry at join node is at most $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$.

The time to compute an entry $c[t,Y,k]$ is at most $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$. Moreover, the number of entries is bounded by $|V(T)| \cdot k \cdot n \in n^{\mathcal{O}(1)}$. Thus, the running time of the algorithm is bounded by $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$. ◀

▶ **Theorem 29.** CCBM *admits an* FPT *algorithm running in time $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$.*

4.2 FPT algorithm for Chordal Conflict Matching.

We design an FPT algorithm for Chordal Conflict Matching, using the algorithm for CCBM (Theorem 29). Let (G, H, k) be an instance of CF-MM, where H is a chordal graph and G is a graph on n vertices. We assume that G is a graph on vertex set $[n]$, which can easily be achieved by renaming vertices.

The algorithm starts by computing an $(n, 2k)$-universal set $\mathcal{F}$, using Proposition 18. For each set $A \in \mathcal{F}$, the algorithm constructs an instance $I_A = (G_A, L_A, R_A, H_A, k)$ of CCBM as follows. We have $V(G_A) = V(G)$, $L_A = A$, $R = V(G) \setminus A$, $E(G_A) = \{uv \in E(G) \mid u \in L_A, v \in R_A\}$, and $H_A = H[E(G_A)]$. Note that H_A is a chordal graph because chordal graphs are closed under induced subgraphs and disjoint unions. The algorithm decides the instance I_A using Theorem 29, for each $A \in \mathcal{F}$. The algorithm outputs yes if and only if there is $A \in \mathcal{F}$, such that I_A is a yes instance of CCBM.

▶ **Theorem 30.** *The algorithm presented for* CF-MM *is correct, Moreover, it runs in time* $2^{\mathcal{O}(\omega k)} k^{\mathcal{O}(\log k)} n^{\mathcal{O}(1)}$*, where* $\omega < 2.373$ *is the exponent of matrix multiplication and* n *is the number of vertices in the input graph.*

Proof. Let (G, H, k) be an instance of CF-MM, where H is a chordal graph and G is a graph on vertex set $[n]$. Clearly, if the algorithm outputs yes, then indeed (G, H, k) is a yes instance of CF-MM. Next, we argue that if (G, H, k) is a yes instance of CF-MM then the algorithm returns yes. Suppose there is a solution $M \subseteq E(G)$ to CF-MM in (G, H, k). Let $S = \{i, j \mid ij \in M\}$, and $L = \{i \mid \text{there is } j \in [n] \text{ such that } ij \in M \text{ and } i < j\}$. Observe that $|S| = 2k$. Since $\mathcal{F}$ is an $(n, 2k)$-universal set, there is $A \in \mathcal{F}$ such that $A \cap S = L$. Note that S is a solution to CCBM in I_A. This together with Theorem 29 implies that the algorithm will return yes as output.

Next, we prove the claimed running time of the algorithm. The algorithm computes $(n, 2k)$-universal set of size $\mathcal{O}(2^{2k} k^{\mathcal{O}(\log k)} \log n)$, in time $\mathcal{O}(2^{2k} k^{\mathcal{O}(\log k)}\, n \log n)$, using Proposition 18. Then for each $A \in \mathcal{F}$, the algorithm creates an instance I_A of CCBM in polynomial time. Furthermore, it resolves the I_A of CCBM in time $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$ using Theorem 29. Hence, the running time of the algorithm is bounded by $2^{\mathcal{O}(\omega k)} k^{\mathcal{O}(\log k)} n^{\mathcal{O}(1)}$. ◀

5 FPT algorithms for CF-MM and CF-SP with matroid constraints

In this section, we study the problems CF-MM and CF-SP, where the conflicting condition is being an independent set in a (representable) matroid. Due to technical reasons (which will be clear later) for the above variant of CF-MM, we will only consider the case when the matroid is repsesentable over $\mathbb{Q}$ (the field of rationals).

5.1 FPT algorithm for Matroid CF-MM

We study a variant of the problem CF-MM, where the conflicting condition is being an independent set in a matroid representable over $\mathbb{Q}$. We call this variant of CF-MM as Rational Matroid CF-MM (Rat Mat CF-MM, for short), which is formally defined below.

599 Rational Matroid CF-MM (Rat Mat CF-MM) **Parameter:** k
Input: A graph G, a matrix $A_{\mathcal{M}}$ (representing a matroid $\mathcal{M}$ over $\mathbb{Q}$) with columns indexed by $E(G)$, and an integer k.
Question: Is there a matching $M \subseteq E(G)$ of size at most k, such that the set of columns in M are linearly independent (over $\mathbb{Q}$)?

We design an FPT algorithm for Rat Mat CF-MM. Towards designing an algorithm for Rat Mat CF-MM, we first give an FPT algorithm for a restricted version of Rat Mat CF-MM, where the graph in which we want to compute a matching is a bipartite graph. We call this variant of Rat Mat CF-MM as Rat Mat CF-Bipartite Matching (Rat Mat CF-BM). We then employ the algorithm for Rat Mat CF-BM to design an FPT algorithm for Rat Mat CF-MM.

5.1.1 FPT algorithm for Rat Mat CF-BM

We design an FPT algorithm for the problem Rat Mat CF-BM, where the conflicting condition is being an independent set in a matroid (representable over $\mathbb{Q}$) and the graph where we want to compute a matching is a bipartite graph. This problem is formally defined below.

> Rat Mat CF-Bipartite Matching (Rat Mat CF-BM) **Parameter:** k
> **Input:** A bipartite graph $G = (V, E)$ with vertex bipartition L, R, a matrix $A_{\mathcal{M}}$ (representing a matroid $\mathcal{M}$ over $\mathbb{Q}$) with columns indexed by E, and an integer k.
> **Question:** Is there a matching $M \subseteq E$ of size k in G, such that the set of columns in M are linearly independent (over $\mathbb{Q}$)?

Our algorithm takes an instance of Rat Mat CF-BM and generates an instance of 3-Matroid Intersection, and then employs the known algorithm for 3-Matroid Intersection to resolve the instance. In the following, we formally define the problem 3-Matroid Intersection.

> 3-Matroid Intersection **Parameter:** k
> **Input:** Matrices $A_{\mathcal{M}_1}, A_{\mathcal{M}_2}$, and $A_{\mathcal{M}_3}$ over $\mathbb{F}$ (representing matroids $\mathcal{M}_1, \mathcal{M}_2$, and $\mathcal{M}_3$, respectively, on the same ground set E) with columns indexed by E, and an integer k.
> **Question:** Is there a set $M \subseteq E$ of size k, such that M is independent in each $\mathcal{M}_i$, for $i \in [3]$?

Before moving further, we briefly explain why we needed an additional constraint that the input matrix is representable over $\mathbb{Q}$. Firstly, we will be using partition matroids which are representable only on fields of large enough size, and we want all the matroids, i.e. the one which is part of the input and the ones that we create, to be representable over the same field. Secondly, the algorithmic result (with the desired running time) we use for 3-Matroid Intersection works only for certain types of fields.

Next, we state an algorithmic result regarding 3-Matroid Intersection [24], which is be used by the algorithm. We note that we only state a restricted form of the algorithmic result for 3-Matroid Intersection in [24], which is enough for our purpose.

▶ **Proposition 31** (Proposition 4.8 [24] (partial))**.** 3-Matroid Intersection *can be solved in $\mathcal{O}(2^{3\omega k}\|A_M\|^{\mathcal{O}(1)})$ time, when the matroids are represented over $\mathbb{Q}$.*

We are now ready to prove the desired result.

▶ **Theorem 32.** Rat Mat CF-BM *can be solved in $\mathcal{O}(2^{3\omega k}\|A_M\|^{\mathcal{O}(1)})$ time.*

Proof. Let $(G = (V, E), L, R, A_{\mathcal{M}}, k)$ be an instance of Rat Mat CF-BM, where the matrix $A_{\mathcal{M}}$ represents a matroid $\mathcal{M} = (E, \mathcal{I})$ over $\mathbb{Q}$. Let $\mathcal{M}_L = (E, \mathcal{I}_L), \mathcal{M}_R = (E, \mathcal{I}_R)$ be the partition matroids as defined in Section 4. Next we compute matrix representations $A_{\mathcal{M}_L}$ and $A_{\mathcal{M}_R}$ of matroids $\mathcal{M}_L, \mathcal{M}_R$, respectively, using Proposition 8. Now, we solve 3-Matroid Intersection on the instance $(\mathcal{M}, A_{\mathcal{M}_L}, A_{\mathcal{M}_R}, k)$ (over $\mathbb{Q}$) using Proposition 31, and return the same answer, as returned by the algorithm in it. The correctness follows directly from the following. $S \subseteq E$ is a matching in G if and only if S is an independent set in $\mathcal{M}_L$ and $\mathcal{M}_R$, that is $S \in \mathcal{I}_L \cap \mathcal{I}_R$. The claimed running time follows from Proposition 8 and Proposition 31. ◀

5.1.2 FPT algorithm for Rat Mat CF-MM

We design an FPT algorithm for Rat Mat CF-MM, using the algorithm for Rat Mat CF-BM (Theorem 29). Let $(G, A_{\mathcal{M}}, k)$ be an instance of Rat Mat CF-MM, where the matrix $A_{\mathcal{M}}$ represents a matroid $\mathcal{M} = (E, \mathcal{I})$ over $\mathbb{Q}$. We assume that G is a graph with the vertex set $[n]$, which can easily be achieved by renaming vertices.

The algorithm starts by computing an $(n, 2k)$-universal set $\mathcal{F}$, using Proposition 18. For each set $X \in \mathcal{F}$, the algorithm constructs an instance $I_X = (G_X, L_X, R_X, A_{\mathcal{M}}, k)$ of Rat Mat CF-BM as follows. We have $V(G_X) = V(G)$, $L_X = X$, $R = V(G) \setminus X$, $E(G_X) = \{uv \in E(G) \mid u \in L_X, v \in R_X\}$. The algorithm decides the instance I_X using Theorem 32, for each $X \in \mathcal{F}$. The algorithm outputs yes if and only if there is $X \in \mathcal{F}$, such that I_X is a yes instance of Rat Mat CF-BM.

▶ **Theorem 33.** *The algorithm presented for* Rat Mat CF-MM *is correct. Moreover, it runs in time* $\mathcal{O}(2^{(3\omega+2)k} k^{\mathcal{O}(\log k)} \|A_M\|^{\mathcal{O}(1)} n^{\mathcal{O}(1)})$.

Proof. Let $(G, A_{\mathcal{M}}, k)$ be an instance of Rat Mat CF-MM, where matrix $A_{\mathcal{M}}$ represent a matroid $\mathcal{M} = (E, \mathcal{I})$ over field $\mathbb{F}$. Clearly, if the algorithm outputs yes, then indeed $(G, A_{\mathcal{M}}, k)$ is a yes instance of Rat Mat CF-MM. Next, we argue that if $(G, A_{\mathcal{M}}, k)$ is a yes instance of Rat Mat CF-MM then the algorithm returns yes. Suppose there is a solution $M \subseteq E(G)$ to Rat Mat CF-MM in $(G, A_{\mathcal{M}}, k)$. Let $S = \{i, j \mid ij \in M\}$, and $L = \{i \mid \text{there is } j \in [n] \text{ such that } ij \in M \text{ and } i < j\}$. Observe that $|S| = 2k$. Since $\mathcal{F}$ is an $(n, 2k)$-universal set, there is $X \in \mathcal{F}$ such that $X \cap S = L$. Note that S is a solution to Rat Mat CF-BM in I_X. This together with Theorem 32 implies that the algorithm will return yes as the output.

Next, we prove the claimed running time of the algorithm. The algorithm computes $(n, 2k)$-universal set of size $\mathcal{O}(2^{2k} k^{\mathcal{O}(\log k)} \log n)$, in time $\mathcal{O}(2^{2k} k^{\mathcal{O}(\log k)}\ n \log n)$, using Proposition 18. Then for each $X \in \mathcal{F}$, the algorithm creates an instance I_X of Rat Mat CF-BM in polynomial time. Furthermore, it resolves the I_X of Rat Mat CF-BM in time $\mathcal{O}(2^{3\omega k} \|A_M\|^{\mathcal{O}(1)})$ using Theorem 32. Hence, the running time of the algorithm is bounded by $\mathcal{O}(2^{(3\omega+2)k} k^{\mathcal{O}(\log k)} \|A_M\|^{\mathcal{O}(1)} n^{\mathcal{O}(1)})$. ◀

5.2 FPT algorithm for Matroid CF-SP

In this section, we design an FPT algorithm for Matroid CF-SP. In the following, we formally define the problem Matroid CF-SP.

Matroid CF-SP	**Parameter:** k
Input: A graph G, (distinct) vertices $s, t \in V(G)$, a matrix $A_{\mathcal{M}}$ (representing a matroid $\mathcal{M}$, over a field $\mathbb{F}$) with columns indexed by $E(G)$, and an integer k.	
Question: Is there a set $M \subseteq E(G)$ of size at most k, such that there is an st-path in $G[M]$ and the set of columns in M are linearly independent (over $\mathbb{F}$)?	

Our algorithm is based on a dynamic programming over representative families. Let $(G, s, t, A_{\mathcal{M}}, k)$ be an instance of Matroid CF-SP. Before moving to the description of the algorithm, we need to define some notations.

Algorithm 1: Alg-Mat-CF-SP

Input: A graph G, (distinct) vertices $s, t \in V(G)$, a matrix $A_{\mathcal{M}}$ (over $\mathbb{F}$), and an integer k.

Output: If there is $M \subseteq E(G)$ of size at most k, such that there is an $s - t$ path in $G[M]$ and the set of columns in M are linearly independent (over $\mathbb{F}$) then yes. Otherwise, no.

for *each* $v \in V(G) \setminus \{s\}$ **do**
  if $sv \in E(G)$ **then** $\mathcal{P}^1_{sv} = \{sv\}$;
  else $\mathcal{P}^1_{sv} = \emptyset$;
  for $q = 0$ *to* $k - 1$ **do**
    Set $\widehat{\mathcal{P}}^{1q}_{sv} = \mathcal{P}^1_{sv}$;
  end
end
for $p = 2$ *to* k **do**
  for $q = 0$ *to* $k - p$ **do**
    for *each* $v \in V(G) \setminus \{s\}$ **do**
      Let $\widetilde{\mathcal{P}}^{pq}_{sv} = \cup_{wv \in E(G)} \widehat{\mathcal{P}}^{(p-1)(q+1)}_{sw} \star \{\{wv\}\}$;
      Compute $\widehat{\mathcal{P}}^{pq}_{sv} \subseteq^{k-p}_{\mathsf{rep}} \widetilde{\mathcal{P}}^{pq}_{sv}$ using Theorem 14;
    end
  end
end
for $p = 1$ *to* k **do**
  for $q = 0$ *to* $k - p$ **do**
    if $\widehat{\mathcal{P}}^{pq}_{st} \neq \emptyset$ **then**
      return yes;
  end
end
return no;

For distinct vertices $u, v \in V(G)$ and an integer p, we define the following.

$$\begin{aligned}\mathcal{P}^p_{uv} = &\{X \subseteq E(G) \mid |X| = p, \text{ there is a } uv\text{-path in } G[X] \text{ containing all edges} \\ &\text{in } X, \text{ and } X \in \mathcal{I}\}\end{aligned} \quad (4)$$

By the definition of convolution of sets, it is easy to see that

$$\mathcal{P}^p_{uv} = \bigcup_{wv \in E(G)} \mathcal{P}^{p-1}_{uw} \star \{\{wv\}\}$$

Now we are ready to describe our algorithm Alg-Mat-CF-SP for Matroid CF-SP. We aim to store, for each $v \in V(G) \setminus \{s\}$, $p \leq k$, and $q \leq k - p$, a q-representative set $\widehat{\mathcal{P}}^{pq}_{sv}$, of $\mathcal{P}^p_{sv}$, of size $\binom{p+q}{q}$. Notice that for each $v \in V(G) \setminus \{s\}$, we can compute $\mathcal{P}^1_{sv}$ in polynomial time, since $\mathcal{P}^1_{sv} = \{sv\}$ if $sv \in E(G)$, and is empty otherwise. Moreover, since $|\mathcal{P}^1_{sv}| \leq 1$, therefore, we can set $\widehat{\mathcal{P}}^{1q}_{sv} = \mathcal{P}^1_{sv}$, for each $q \leq k - 1$. Next, we iteratively compute, for each $p \in \{2, 3, \cdots, k\}$, in increasing order, for each $q \leq k - p$, a q-representative $\widehat{\mathcal{P}}^{pq}_{sv}$, of $\mathcal{P}^p_{sv}$. The algorithm Alg-Mat-CF-SP is given in Algorithm 1.

Next, we prove a lemma which will be useful in establishing the correctness of Alg-Mat-CF-SP.

▶ **Lemma 34.** *For each $p \in [k]$, $q \in [0, k-p]$, and $v \in V(G) \setminus \{s\}$, the family $\widehat{\mathcal{P}}_{sv}^{pq}$ computed by* Alg-Mat-CF-SP *is a q-representative of $\mathcal{P}_{sv}^{p}$, and is of size at most $\binom{p+q}{q}$. Moreover, the algorithm computes all sets in $\{\widehat{\mathcal{P}}_{sv}^{pq} \mid p \in [k], q \in [0, k-p], v \in V(G) \setminus \{s\}\}$ in time $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$.*

Proof. We prove the claim by induction on p. Consider $v \in V(G) \setminus \{s\}$. For $p = 1$, the set $\mathcal{P}_{sv}^{1} = \{sv\}$ if $sv \in E(G)$, and is empty otherwise. Moreover, for each $q \in [0, k-1]$, Alg-Mat-CF-SP sets $\widehat{\mathcal{P}}_{sv}^{1q} = \mathcal{P}_{sv}^{1}$. Hence, for each $q \in [0, k-1]$, we have $\widehat{\mathcal{P}}_{sv}^{1q} \subseteq_{\mathsf{rep}}^{q} \mathcal{P}_{sv}^{1}$. Moreover, the set $\widehat{\mathcal{P}}_{sv}^{1q}$ is computed by the algorithm in polynomial time.

For induction hypothesis, we assume that for $t \in \mathbb{N}_{\geq 1}$, for each $p \leq t$, $q \in [0, k-p]$, and $v \in V(G) \setminus \{s\}$, we have $\widehat{\mathcal{P}}_{sv}^{pq} \subseteq_{\mathsf{rep}}^{q} \mathcal{P}_{sv}^{p}$. Next, consider $p = t+1$, $q \in [0, k-(t+1)]$, and $v \in V(G) \setminus \{s\}$. The step of the algorithm, where it computes $\widehat{\mathcal{P}}_{sv}^{(t+1)q}$, it has already computed (correctly), for each $p \leq t$, $q \in [0, k-p]$, and $u \in V(G) \setminus \{s\}$, the set $\widehat{\mathcal{P}}_{su}^{pq} \subseteq_{\mathsf{rep}}^{q} \mathcal{P}_{su}^{p}$. This follows from the iteration of the algorithm over p from 1 to k in increasing order at Step 6 (and Step 1). The algorithm sets $\widetilde{\mathcal{P}}_{sv}^{(t+1)q} = \cup_{wv \in E(G)} \widehat{\mathcal{P}}_{sw}^{(t)(q+1)} \star \{\{wv\}\}$, and then sets $\widehat{\mathcal{P}}_{sv}^{(t+1)q}$ to be the q-representative set of $\widetilde{\mathcal{P}}_{sv}^{(t+1)q}$ returned by Theorem 14, which is of size at most $\binom{t+1+q}{t+1}$. If we show that $\widetilde{\mathcal{P}}_{sv}^{(t+1)q} \subseteq_{\mathsf{rep}}^{q} \mathcal{P}_{sv}^{t+1}$, then by Proposition 12 it will follow that $\widehat{\mathcal{P}}_{sv}^{(t+1)q} \subseteq_{\mathsf{rep}}^{q} \mathcal{P}_{sv}^{t+1}$. But $\widetilde{\mathcal{P}}_{sv}^{(t+1)q} \subseteq_{\mathsf{rep}}^{q} \mathcal{P}_{sv}^{t+1}$ follows from Lemma 15 and Proposition 13. Also, note that each entry can be computed in time $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$. ◀

Using Lemma 34, we obtain the following theorem.

▶ **Theorem 35.** *The algorithm* Alg-Mat-CF-SP *is correct. Moreover, it runs in time $\mathcal{O}(2^{\mathcal{O}(\omega k)} n^{\mathcal{O}(1)})$.*

Proof. Let $(G, s, t, A_{\mathcal{M}}, k)$ be an instance of Matroid CF-SP. We claim that $(G, s, t, A_{\mathcal{M}}, k)$ is a yes instance of Matroid CF-SP if and only if Alg-Mat-CF-SP outputs yes. In the forward direction, let $(G, s, t, A_{\mathcal{M}}, k)$ be a yes instance of Matroid CF-SP. Since, using Lemma 34, Alg-Mat-CF-SP computes a q-representative of $\mathcal{P}_{sv}^{p}$ of size at most $\binom{p+q}{q}$, for each $p \in [k]$, $q \in [0, k-p]$, and $v \in V(G) \setminus \{s\}$, therefore, the algorithm also computes a q-representative family for $\mathcal{P}_{st}^{k}$. By the definition of representative set and construction of our family $\mathcal{P}_{st}^{k}$, $\widehat{\mathcal{P}}_{st}^{k}$ also contains a $s-t$ path and hence, the algorithm outputs yes. In the reverse direction, if the algorithm outputs yes then by construction of family $\widehat{\mathcal{P}}_{st}^{k}$, if $P \in \widehat{\mathcal{P}}_{st}^{k}$, then it is a conflict-free $s-t$ path in G. This completes the correctness of our algorithm. Moreover, the running time bound of the algorithm follows from Lemma 34. ◀

Theorem 35 will also result into an FPT algorithm for CF-SP when the conflict graph is a cluster graph.

▶ **Corollary 36.** Conflict Free Shortest Path *parameterized by the solution size is* FPT*, when the conflict graph is a cluster graph.*

Proof. Let (G, H, k) be an instance of CF-SP, where H is a cluster graph. We construct a partition matroid, $\mathcal{M}_H = (U, \mathcal{I})$, corresponding to graph H as follows. We define ground set as $U = V(H)$. Now, partition U as $U_i = V(C_i)$, for each clique C_i in H and $a_i = 1$, for $U_i \in U$. By the construction of $\mathcal{M}_H$, we have for $S \subseteq V(H)$, S is an independent set in H if and only if S is independent in $\mathcal{M}_H$. Next, we compute a matrix M representing $\mathcal{M}_H$, using Proposition 8 in polynomial time.Now we use Alg-Mat-CF-SP with input (G, M, k), and return the same output. The correctness of our algorithm follows from correctness of the algorithm Alg-Mat-CF-SP (Theorem 35), and by construction of the instance (G, M, k). Note that the matrix M representing $\mathcal{M}_H$ can be computed in polynomial time. This together with Theorem 35 implies the claimed running time bound, This concludes the proof. ◀

6 FPT Algorithm for d-degenerate Conflict Graphs

In this section, we show that CF-MM and CF-SP both are in FPT, when the conflict graph H is a d-degenerate graphs. These algorithms are based on the notion of independence covering family, which was introduced in [25].

Before moving onto description of our algorithms, we define the notion of independence covering family.

▶ **Definition 37** ([25]). For a graph $H^\star$ and an integer k, a *k-independence covering family*, $\mathscr{I}(H^\star, k)$, is a family of independent sets in $H^\star$ such that for any independent set I' in $H^\star$ of size at most k, there is a set $I \in \mathscr{I}(H^\star, k)$ such that $I' \subseteq I$.

Our algorithms rely on the construction of k-independence covering family, for a family of graphs. But before dwelling into these details, we first design an algorithm for an annotated version of the CF-MM and CF-SP problems, which we call Annotated CF-MM and Annotated CF-SP, respectively. In the Annotated CF-MM (Annotated CF-SP) problem, the input to CF-MM (CF-SP) is annotated with a k-independence covering family.

6.1 Algorithms for Annotated CF-MM and Annotated CF-SP

In this section, we study the problems Annotated CF-MM and Annotated CF-SP, which are formally defined below.

Annotated CF-MM **Parameter:** $k + |\mathcal{F}|$
Input: A graph $G = (V, E)$, a conflict graph $H = (E, E')$, an integer k, and a k-independence covering family $\mathcal{F}$ of H.
Question: Is there a matching $M \subseteq E$ of size k in G such that M is an independent set in H?

Annotated CF-SP **Parameter:** $k + |\mathcal{F}|$
Input: A graph $G = (V, E)$, (distinct) vertices $s, t \in V$, a conflict graph $H = (E, E')$, an integer k, and a k-independence covering family $\mathcal{F}$ of H.
Question: Is there a set $M \subseteq E$ of size at most k, such that there is an $s - t$ path in $G[M]$ and M is an independent set in H?

The algorithm that we design for Annotated CF-MM runs in time polynomial in the size of the input. We give the algorithm Alg-CF-MM (Alg-CF-SP) (Algorithm 2) for Annotated CF-MM (Annotated CF-SP).

In the following lemma we prove the correctness of Alg-CF-MM (Alg-CF-SP).

▶ **Lemma 38.** *The algorithm* Alg-CF-MM (Alg-CF-SP) *is correct. Moreover, the algorithm runs in time polynomial in the size of the input.*

Proof. Let $(G, (s, t), H, k, \mathcal{F})$ be an instance of Annotated CF-MM (Annotated CF-SP). We show that $(G, (s, t), H, k, \mathcal{F})$ is a yes instance of Annotated CF-MM (Annotated CF-SP) if and only if Alg-CF-MM (Alg-CF-SP) outputs yes.

Note that the reverse direction easily follows from the fact that $\mathcal{F}$ is a family of independent sets in H. Therefore, we only need to prove the forward direction. In the forward direction, let $(G, (s, t), H, k, \mathcal{F})$ be a yes instance of Annotated CF-MM (Annotated CF-SP) and S be one of its solution. Since $\mathcal{F}$ is a k-independence covering family, there is $I \in \mathcal{F}$ such that $S \subseteq I$ (see Definition 37). Hence, in the iteration where the algorithm considers I in its for loop, the graph G_I has S as a matching (there is an $s - t$ path in $G_I[S]$). Therefore, the algorithm outputs yes at this iteration.

Algorithm 2: Alg-CF-MM (Alg-CF-SP)

Input: A graph G,((distinct) vertices $s, t \in V(G)$), a conflict graph H, an integer k, and a k-independence covering family $\mathcal{F}$ of H.

Output: If there a set $M \subseteq E$ of size k in G such that M is a matching in G (there is an $s-t$ path in $G[M]$) and M is an independent set in H, then yes, and no otherwise.

```
for each I ∈ F do
    Let G_I be the graph with V(G_I) = V(G) and E(G_I) = I ;
    if G_I has a matching (path) of size k then
        return yes;
end
return no ;
```

The running time analysis follows from the fact that maximum matching (shortest path) can be computed in polynomial time [12, 27]([7, 3]). ◀

Next, we use Alg-CF-MM (Alg-CF-SP) together with Independence Covering Lemma of [25] to obtain algorithms for CF-MM (CF-SP) when the conflict graph is d-degenerate or nowhere dense graph. Towards this we state some lemmata from [25] that we use in our algorithms.

▶ **Proposition 39.** [25, Lemma 1.1] *There is a randomized algorithm running in polynomial time, that given a d-degenerate graph $H^\star$ and an integer k as input, outputs an independent set I, such that for every independent set I' of size at most k in graph $H^\star$, the probability that $I' \subseteq I$ is at least $(\binom{k(d+1)}{k} \cdot k(d+1))^{-1}$.*

▶ **Proposition 40.** [25, Lemmas 3.2 and 3.3] *There are two deterministic algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, which given a d-degenerate graph $H^\star$ and an integer k, output independence covering families $\mathscr{I}_1(H^\star, k)$ and $\mathscr{I}_2(H^\star, k)$, respectively, such that the following conditions are satisfied.*

- *$\mathcal{A}_1$ runs in time $\mathcal{O}(|\mathscr{I}_1(H^\star, k)| \cdot (n+m))$, where $|\mathscr{I}_1(H^\star, k)| = \binom{k(d+1)}{k} \cdot 2^{o(k(d+1))} \cdot \log n$.*
- *$\mathcal{A}_2$ runs in time $\mathcal{O}(|\mathscr{I}_2(H^\star, k)| \cdot (n+m))$, where $|\mathscr{I}_2(H^\star, k)| = \binom{k^2(d+1)^2}{k} \cdot (k(d+1))^{\mathcal{O}(1)} \cdot \log n$.*

Next, using Proposition 39 and 40, together with Alg-CF-MM (Alg-CF-SP), we obtain randomized and deterministic algorithms, respectively for CF-MM (CF-SP), when the conflict graph is a d-degenerate graph.

▶ **Theorem 41.** *There is a randomized algorithm, which given an instance (G, H, k) of* CF-MM(CF-SP)*, where H is a d-degenerate graph, in time $\binom{k(d+1)}{k} \cdot k(d+1) \cdot n^{\mathcal{O}(1)}$, either reports a failure or correctly outputs that the input is a* yes *instance of* CF-MM(CF-SP)*. Moreover, if the input is a* yes *instance of* CF-MM(CF-SP)*, then the algorithm outputs correct answer with a constant probability.*

Proof. Let $(G, (s,t), H, k)$ be an instance CF-MM (CF-SP), where H is a d-degenerate graph.

We repeat the following procedure $(\binom{k(1+d)}{k} \cdot k(d+1))$ many times.

1. The algorithm computes an independent set I in (H, k) using Proposition 39.
2. The algorithm calls Alg-CF-MM (Alg-CF-SP) with input $(G, (s,t)H, k, \{I\})$.

The algorithm outputs yes, if in one of the calls to Alg-CF-MM (Alg-CF-SP), it receives a yes. Otherwise, the algorithm outputs no. The running time analysis of the above procedure follows from Proposition 39 and Lemma 38. Also, given a yes instance, the guarantee on success probability follows from Proposition 39, the number of repetitions, and Lemma 38. Moreover, from Lemma 38 the yes output returned by the algorithm is indeed the correct output to CF-MM(CF-SP)for the given instance. This concludes the proof. ◀

▶ **Theorem 42.** CF-MM (CF-SP) *admits a deterministic algorithm running in time* min $\left\{\binom{k(d+1)}{k} \cdot 2^{o(k(d+1))} \cdot \log n, \binom{k^2(d+1)^2}{k} \cdot (k(d+1))^{\mathcal{O}(1)} \cdot \log n\right\} \cdot n^{\mathcal{O}(1)}$*, when the conflict graph is a d-degenerate graph.*

Proof. Let $(G,(s,t),H,k)$ be an instance CF-MM (CF-SP), where H is a d-degenerate graph. The algorithm starts by computing a k-independence covering family $\mathscr{I}(H,k)$ of H, using Proposition 40. Next, we call Alg-CF-MM (Alg-CF-SP) with the input $(G,(s,t),H,k,$ $\mathscr{I}(H,k))$. The correctness and running time analysis of the above procedure follows from Proposition 40 and Lemma 38. This completes the proof. ◀

References

1. Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, Daniel Lokshtanov, and Saket Saurabh. Conflict free feedback vertex set: A parameterized dichotomy. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS*, pages 53:1–53:15, 2018.
2. Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, Pranabendu Misra, and Saket Saurabh. Exploring the kernelization borders for hitting cycles. In *13th International Symposium on Parameterized and Exact Computation, IPEC*, pages 14:1–14:14, 2018.
3. Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.
4. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
5. Andreas Darmann, Ulrich Pferschy, and Joachim Schauer. Determining a minimum spanning tree with disjunctive constraints. In *International Conference on Algorithmic DecisionTheory (ADT)*, pages 414–423, 2009.
6. Andreas Darmann, Ulrich Pferschy, Joachim Schauer, and Gerhard J. Woeginger. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726–1735, 2011.
7. Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
8. Rodney G. Downey and Michael R. Fellows. Fixed parameter tractability and completeness. In *Complexity Theory: Current Research*, pages 191–225. Cambridge University Press, 1992.
9. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
10. Leah Epstein, Lene M. Favrholdt, and Asaf Levin. Online variable-sized bin packing with conflicts. *Discrete Optimization*, 8(2):333–343, 2011.
11. Guy Even, Magnús M. Halldórsson, Lotem Kaplan, and Dana Ron. Scheduling with conflicts: online and offline algorithms. *Journal of Scheduling*, 12(2):199–224, 2009.
12. Shimon Even and Oded Kariv. An $O(n^{2.5})$ algorithm for maximum matching in general graphs. In *Foundations of Computer Science (FOCS)*, pages 100–112, 1975.
13. J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag, Secaucus, NJ, USA, 2006.
14. Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *Journal of the ACM*, 63(4):29:1–29:60, 2016.

15 Fedor V Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *Symposium on Discrete Algorithms (SODA)*, pages 142–151, 2014.

16 Harold N. Gabow, Shachindra N Maheshwari, and Leon J. Osterweil. On two problems in the generation of program test paths. *IEEE Transactions on Software Engineering*, 2(3):227–231, 1976.

17 Fănic˘ a Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

18 Michel Gendreau, Gilbert Laporte, and Frédéric Semet. Heuristics and lower bounds for the bin packing problem with conflicts. *Computers & OR*, 31(3):347–358, 2004.

19 Pallavi Jain, Lawqueen Kanesh, and Pranabendu Misra. Conflict free version of covering problems on graphs: Classical and parameterized. In *Computer Science - Theory and Applications - 13th International Computer Science Symposium in Russia (CSR)*, pages 194–206, 2018.

20 Klaus Jansen. An approximation scheme for bin packing with conflicts. *Journal of combinatorial optimization*, 3(4):363–377, 1999.

21 Minghui Jiang. On the parameterized complexity of some optimization problems related to multiple-interval graphs. *Theoretical Computer Science*, 411(49):4253–4262, 2010.

22 Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

23 KW Krause, MA Goodwin, and RW Smith. *Optimal software test planning through automated network analysis*. TRW Systems Group, 1973.

24 Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 922–934, 2015.

25 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Covering small independent sets and separators with applications to parameterized algorithms. In *Symposium on Discrete Algorithms (SODA)*, pages 2785–2800, 2018.

26 Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009.

27 Silvio Micali and Vijay V Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Foundations of Computer Science (FOCS)*, pages 17–27, 1980.

28 Moni Naor, Leonard J Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Foundations of Computer Science (FOCS)*, pages 182–191, 1995.

29 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.

30 James G Oxley. *Matroid theory*, volume 3. Oxford University Press, 2006.

31 Ulrich Pferschy and Joachim Schauer. The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, 13(2):233–249, 2009.

32 Ulrich Pferschy and Joachim Schauer. The maximum flow problem with conflict and forcing conditions. In *Network Optimization*, pages 289–294. Springer, 2011.

33 Ulrich Pferschy and Joachim Schauer. The maximum flow problem with disjunctive constraints. *Journal of Combinatorial Optimization*, 26(1):109–119, 2013.

34 Ulrich Pferschy and Joachim Schauer. Approximation of knapsack problems with conflict and forcing graphs. *Journal of Combinatorial Optimization*, 33(4):1300–1323, 2017.

35 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Symposium on Theory of Computing (STOC)*, pages 887–898, 2012.