# Representative Families of Product Families

FEDOR V. FOMIN and DANIEL LOKSHTANOV, University of Bergen, Norway
FAHAD PANOLAN, The Institute of Mathematical Sciences, HBNI, India
SAKET SAURABH, The Institute of Mathematical Sciences, HBNI, India and University of Bergen, Norway

A subfamily $\mathcal{F}'$ of a set family $\mathcal{F}$ is said to *q-represent* $\mathcal{F}$ if for every $A \in \mathcal{F}$ and $B$ of size $q$ such that $A \cap B = \emptyset$ there exists a set $A' \in \mathcal{F}'$ such that $A' \cap B = \emptyset$. Recently, we provided an algorithm that for a given family $\mathcal{F}$ of sets of size $p$ together with an integer $q$, efficiently computes a $q$-representative family $\mathcal{F}'$ of $\mathcal{F}$ of size approximately $\binom{p+q}{p}$. In this paper, we consider the efficient computation of $q$-representative families for *product* families $\mathcal{F}$. A family $\mathcal{F}$ is a product family if there exist families $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{F} = \{A \cup B \ : \ A \in \mathcal{A}, B \in \mathcal{B}, A \cap B = \emptyset\}$. Our main technical contribution is an algorithm which given $\mathcal{A}$, $\mathcal{B}$ and $q$ computes a $q$-representative family $\mathcal{F}'$ of $\mathcal{F}$. The running time of our algorithm is *sublinear* in $|\mathcal{F}|$ for many choices of $\mathcal{A}$, $\mathcal{B}$ and $q$ which occur naturally in several dynamic programming algorithms. We also give an algorithm for the computation of $q$-representative families for product families $\mathcal{F}$ in the more general setting where $q$-representation also involves independence in a matroid in addition to disjointness. This algorithm considerably outperforms the naive approach where one first computes $\mathcal{F}$ from $\mathcal{A}$ and $\mathcal{B}$, and then computes the $q$-representative family $\mathcal{F}'$ from $\mathcal{F}$.

We give two applications of our new algorithms for computing $q$-representative families for product families. The first is a $3.8408^k n^{O(1)}$ deterministic algorithm for the Multilinear Monomial Detection ($k$-MlD) problem. The second is a significant improvement of deterministic dynamic programming algorithms for "connectivity problems" on graphs of bounded treewidth.

CCS Concepts: •**Theory of computation** →**Fixed parameter tractability;**

Additional Key Words and Phrases: matroids, representative families, parameterized algorithms, multi-linear monomial detection, tree-width bounded graphs

**ACM Reference format:**
Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. 2010. Representative Families of Product Families. *ACM Trans. Web* 9, 4, Article 39 (March 2010), 29 pages.
DOI: 0000001.0000001

## 1 INTRODUCTION

Let $M = (E, \mathcal{I})$ be a matroid and let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a family of subsets of $E$ of size $p$. A subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is *q-representative* for $\mathcal{S}$ if for every set $Y \subseteq E$ of size at most $q$, if there is a set $X \in \mathcal{S}$ disjoint from $Y$ with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from $Y$ with $\widehat{X} \cup Y \in \mathcal{I}$.

Preliminary version of this paper appeared in the proceedings of ESA 2014. This work is supported by Rigorous Theory of Preprocessing, ERC Advanced Investigator Grant 267959 and Parameterized Approximation, ERC Starting Grant 306992.
Author's addresses: Fedor V. Fomin, Daniel Lokshtanov, and Fahad Panolan, Department of Informatics, University of Bergen, 5020 Bergen, Norway; Saket Saurabh, Theoretical Computer Science, The Institute of Mathematical Sciences, HBNI, Chennai, 600113, India.

DOI: 0000001.0000001

In other words, if a set $Y$ of size at most $q$ can be extended to an independent set by adding a subset from $\mathcal{S}$, then it also can be extended to an independent set by adding a subset from $\widehat{\mathcal{S}}$ as well. Thus for certain applications the family $\widehat{\mathcal{S}}$ contains the "essential" information about the whole family $\mathcal{S}$ and independent sets of $M$.

The crucial property of representative families used in combinatorics and algorithms, see e.g. [15, Section 9.2.2] and [28, 29], is that for certain matroids the size of a $q$-representative family can be significantly smaller that the size of $\mathcal{S}$ and that such a family can be computed efficiently. By the classic result of Lovász [22], for linear matroids, i.e. matroids representable over a finite field, there exists a representative family $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$ with at most $\binom{p+q}{p}$ sets. However, it is a very non-trivial task of constructing such a representative family efficiently. Monien in [25] provided an algorithm computing a $q$-representative family of size at most $\sum_{i=0}^{q} p^i$ in time $O(pq \cdot \sum_{i=0}^{q} p^i \cdot t)$ for set families, or equivalently for uniform matroids. Marx [23] gave an algorithm, also for uniform matroids, for finding a $q$-representative family of size at most $\binom{p+q}{p}$ in time $O(p^q \cdot t^2)$. For linear matroids, Marx in [24] has shown how Lovász's proof can be transformed into an algorithm computing a $q$-representative family of size at most $\binom{p+q}{p}$ with running time $2^{O(p\log(p+q))} \cdot \binom{p+q}{p}^{O(1)} (||A_M||t)^{O(1)}$, where $||A_M||$ is the size of the input representation matrix of the matroid. Recently, we have shown in [10] how to compute a $q$-representative family with at most $\binom{p+q}{p}$ sets in $O\left(\binom{p+q}{p} tp^\omega + t\binom{p+q}{q}^{\omega-1}\right)$ operations over the field representing the matroid. Here, $\omega < 2.373$ is the matrix multiplication exponent [13, 31]. For the special case of uniform matroids on $n$ elements, we gave a faster algorithm computing a representative family in time $O((\frac{p+q}{q})^q \cdot 2^{o(p+q)} \cdot t \cdot \log n)$. The efficient computations of representative families led to fast deterministic parameterized algorithms for $k$-Path, $k$-Tree, and more generally, for $k$-Subgraph Isomorphism, where the $k$-vertex pattern graph is of constant treewidth in [10].

All currently known algorithms that use fast computation of representative families as a subroutine are based on dynamic programming. It is therefore very tempting to ask whether the computation of representative families can be faster for families that arise naturally in dynamic programs rather than for general families. A class of families which often arises in dynamic programs is the class of *product* families; a family $\mathcal{F}$ is the *product* of $\mathcal{A}$ and $\mathcal{B}$ if $\mathcal{F} = \mathcal{A} \circ \mathcal{B} = \{A \cup B \ : \ A \in \mathcal{A}, B \in \mathcal{B} \wedge A \cap B = \emptyset\}$. Product families naturally appear in dynamic programs where sets represent partial solutions and two partial solutions can be combined if they are disjoint. For an example, in the $k$-Path problem partial solutions are vertex sets of paths starting at a particular root vertex $v$, and two such paths may be combined to a longer path if and only if they are disjoint (except for overlapping at $v$). Many other examples exist—essentially product families can be thought of as a *subset convolution* [2, 3], and the wide applicability of the fast subset convolution technique of Bjorklund et al [4] is largely due to the frequent demand to compute product families in dynamic programs.

**Our results.** Our main technical contributions are two algorithms for the computation of representative families for product families, one for uniform matroids, and one for linear matroids. For uniform matroids we give an algorithm which given an integer $q$ and families $\mathcal{A}$, $\mathcal{B}$ of sets of sizes $p_1$ and $p_2$ over the ground set of size $n$, computes a $q$-representative family $\mathcal{F}'$ of $\mathcal{F}$. The running time of our algorithm is *sublinear* in $|\mathcal{F}|$ for many choices of $\mathcal{A}$, $\mathcal{B}$ and $q$ which occur naturally in several dynamic programming algorithms. For example, let $q, p_1, p_2$ be integers. Let $k = q + p_1 + p_2$ and suppose that we have families $\mathcal{A}$ and $\mathcal{B}$, which are $(k - p_1)$ and $(k - p_2)$-representative families. Then the sizes of these families are roughly $|\mathcal{A}| = \binom{k}{p_1}$ and $|\mathcal{B}| = \binom{k}{p_2}$.

In particular, when $p_1 = p_2 = \lceil k/2 \rceil$ both families are of size roughly $2^k$, and thus the cardinality of $\mathcal{F}$ is approximately $4^k$. On the other hand, for any choice of $p_1$, $p_2$, and $k$, our algorithm outputs a $(k - p_1 - p_2)$-representative family of $\mathcal{F}$ of size roughly $\binom{k}{p_1+p_2}$ in time $3.8408^k n^{O(1)}$. For many choices of $p_1$, $p_2$ and $q$ our algorithm runs significantly faster than $3.8408^k n^{O(1)}$. The expression capturing the running time dependence on $p_1$, $p_2$ and $q$ can be found in Theorem 3.3 and Corollary 3.4.

Our second algorithm is for computing representative families of product families, when the universe is also enriched with a linear matroid. More formally, let $M = (E, \mathcal{I})$ be a matroid and let $\mathcal{A}, \mathcal{B} \subseteq \mathcal{I}$. Then let $\mathcal{F} = \mathcal{A} \bullet \mathcal{B} = \{A \cup B \ : A \cup B \in \mathcal{I}, A \in \mathcal{A}, B \in \mathcal{B} \text{ and } A \cap B = \emptyset\}$. Just as for uniform matroids, a naive approach for computing a representative familiy of $\mathcal{F}$ would be to compute the product $\mathcal{A} \bullet \mathcal{B}$ first and then compute a representative family of the product. The fastest currently known algorithm for computing a representative family is by Fomin et al. [10] and has running time approximately $\binom{p+q}{p}^{\omega-1} |\mathcal{F}|$. We give an algorithm that significantly outperforms the naive approach. An appealing feature of our algorithm is that it works by reducing the computation of a representative family for $\mathcal{F}$ to the computation of represesentative families for many smaller families. Thus an improved algorithm for the computation of representative families for general families will automatically accelerate our algorithm for product families as well. The expression of the running time of our algorithm can be found in Theorem 4.2.

**Applications.** Our first application is a deterministic algorithm for the following parameterized version of multilinear monomial testing.

| | |
|---|---|
| Multilinear Monomial Detection ($k$-MlD) | **Parameter:** $k$ |
| **Input:** An arithmetic circuit $C$ over $\mathbb{Z}^+$ representing a polynomial $P(X)$ over $\mathbb{Z}^+$. | |
| **Question:** Does $P(X)$ construed as a sum of monomials contain a multilinear monomial of degree $k$? | |

This is the central problem in the algebraic approach of Koutis and Williams for designing fast parameterized algorithms [17–19, 30]. The idea behind the approach is to translate a given problem into the language of algebra by reducing it to the problem of deciding whether a constructed polynomial has a multilinear monomial of degree $k$. As it is mentioned implicitly by Koutis in [17], $k$-MlD can be solved in time $(2e)^k n^{O(1)}$, where $n$ is the input length, by making use of color coding. The color coding technique of Alon, Yuster and Zwick [1] is a fundamental and widely used technique in the design of parameterized algorithms. It appeared that most of the problems solvable by making use of color coding can be reduced to a multilinear monomial testing. Williams [30] gave a *randomized* algorithm solving $k$-MlD in time $2^k n^{O(1)}$. The algorithms based on the algebraic method of Koutis-Williams provide a dramatic improvement for a number of fundamental problems [5, 6, 11, 14, 17–19, 30]. See also the recent survey [20].

The advantage of the algebraic approach over color coding is that for a number of parameterized problems, the algorithms based on this approach have much better exponential dependence on the parameter. On the other hand color coding based algorithms admit direct derandomization [1] and are able to handle integer weights with running time overhead poly-logarithmic in the weights. Obtaining deterministic algorithms matching the running times of the algebraic methods, but sharing these nice features of color coding remain a challenging open problem. Our deterministic algorithm for $k$-MlD is the first non-trivial step towards resolving this problem. In fact, our algorithm solves a weighted version of $k$-MlD, where the elements of $X$ are assigned weights and the task is to find a $k$-multilinear term with minimum weight. The running time of our deterministic algorithm is $O(3.8408^k 2^{o(k)} s(C) n \log W \log^2 n)$, where $s(C)$ is the size of the circuit and $W$ is the maximum weight of an element from $X$.

We also provide an algorithm for a more general version of multilinear monomial testing, where variables of a monomial should form an independent set of a linear matroid. The new algorithm can be used as the basic step in solving general optimization problems of finding a subgraph with additional constraints provided in the form of independent sets of some matroids. See, for example, [27].

The second application of our fast computation of representative families is for dynamic programming algorithms on graph of bounded treewidth. It is well known that many intractable problems can be solved efficiently when the input graph has bounded treewidth. Moreover, many fundamental problems like Maximum Independent Set or Minimum Dominating Set can be solved in time $2^{O(t)}n$ [8]. On the other hand, it was believed until very recently that for some "connectivity" problems such as Hamiltonian Cycle or Steiner Tree no such algorithm exists. In their breakthrough paper, Cygan et al. [9] introduced a new algorithmic framework called Cut&Count and used it to obtain $2^{O(t)}n^{O(1)}$ time Monte Carlo algorithms for a number of connectivity problems. Recently, Bodlaender et al. [7] obtained the first deterministic single-exponential algorithms for these problems using two novel approaches. One of the approaches of Bodlaender et al. is based on rank estimations in specific matrices and the second based on matrix-tree theorem and computation of determinants. In [10], Fomin et al. used efficient algorithms for computing representative families of linear matroids to provide yet another approach for single-exponential algorithms on graphs of bounded treewdith.

It is interesting to note that for a number of connectivity problems such as Steiner Tree or Feedback Vertex Set the "bottleneck" of treewidth based dynamic programming algorithms is the *join* operation. For example, as it was shown by Bodlaender et al. in [7], Feedback Vertex Set and Steiner Tree can be solved in time $O\left((1+2^{\omega})^{\mathbf{pw}}\mathbf{pw}^{O(1)}n\right)$ and $O\left((1+2^{\omega+1})^{\mathbf{tw}}\mathbf{tw}^{O(1)}n\right)$, where $\mathbf{pw}$ and $\mathbf{tw}$ are the pathwidth and the treewidth of the input graph. The reason for the difference in the exponents of these two algorithms is due to the cost of the join operation, which is required for treewidth and does not occur for pathwidth. For many computational problems on graphs of bounded treewidth in the join nodes of the decomposition, the family of partial solutions is the product of the families of its children, and we wish to store a representative family (for a graphic matroid) for this product family. Here our second algorithm comes into play. By making use of this algorithm one can obtain faster deterministic algorithms for many connectivity problems. We exemplify this by providing algorithms with running time $O\left((1+2^{\omega-1}\cdot 3)^{\mathbf{tw}}\mathbf{tw}^{O(1)}n\right)$ for Feedback Vertex Set and Steiner Tree.

**Our methods.** Consider a pair of disjoint sets $A$ and $B$, with $|A| = p$ and $|B| = q$. A random coloring which colors each element in $U$ red with probability $\frac{p}{p+q}$ and blue with probability $\frac{q}{p+q}$ will color $A$ red and $B$ blue with probability roughly $\frac{1}{\binom{p+q}{p}}$. Thus a family of slightly more than $\binom{p+q}{p}$ such random colorings will contain, with high probability, for each pair of disjoint sets $A$ and $B$, with $|A| = p$ and $|B| = q$ a function which colors $A$ red and $B$ blue. The fast computation of representative families of Fomin et al. [10] deterministically constructs a collection of colorings which mimics this property of random coloring families. The colorings in the family are used to *witness disjointedness*, since a coloring which colors $A$ red and $B$ blue certifies that $A$ and $B$ are disjoint. In our setting we can use such coloring families both for witnessing disjointedness in the computation of representive sets, and in the computation of $\mathcal{F} = \mathcal{A} \circ \mathcal{B}$. After all, each set in $\mathcal{F}$ is the disjoint union of a set in $\mathcal{A}$ and a set in $\mathcal{B}$. In order to make this idea work we use the deterministic construction of coloring familes given in [10].

For linear matroids, our algorithm computes a representative family $\mathcal{F}'$ of $\mathcal{F} = \mathcal{A} \bullet \mathcal{B}$ as follows. First the family $\mathcal{F}$ is broken up into many smaller families $\mathcal{F}_1, \ldots, \mathcal{F}_t$, and then a representative family $\mathcal{F}_i{}'$ is computed for each $\mathcal{F}_i$. Finally $\mathcal{F}'$ is obtained by computing a representative family of $\bigcup_i \mathcal{F}_i{}'$ using the algorithm of Fomin et al. [10] for computing representative families. The speedup over the naive method is due to the fact that (a) $\bigcup_i \mathcal{F}_i{}'$ is much smaller than $\mathcal{F}$ and (b) each $\mathcal{F}_i$ has a certain structure which ensures better upper bounds on the size of $\mathcal{F}_i{}'$, and allows $\mathcal{F}_i{}'$ to be computed faster.

## 2 PRELIMINARIES

In this section we give various definitions which we make use of in the paper.

**Graphs.** Let $G$ be a graph with vertex set $V(G)$ and edge set $E(G)$. A graph $G'$ is a *subgraph* of $G$ if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. The subgraph $G'$ is called an *induced subgraph* of $G$ if $E(G') = \{uv \in E(G) \mid u, v \in V(G')\}$. In this case, $G'$ is also called the subgraph *induced by* $V(G')$ and denoted by $G[V(G')]$. For a vertex set $S$, by $G \setminus S$ we denote $G[V(G) \setminus S]$, and by $E(S)$ we denote the edge set $E(G[S])$. For an edge set $E'$, we use $G \setminus E'$ to represent the graph with vertex set $V(G)$ and edge set $E(G) \setminus E'$.

**Sets, Functions and Constants.** Let $[n] = \{0, \ldots, n-1\}$. Let $U$ be a set. We use $2^U$, $\binom{U}{i}$ and $\binom{U}{\leq i}$ to denote the family of all subsets of $U$, the family of all subsets of size $i$ of $U$ and the family of all subsets of size at most $i$ of $U$, respectively. A family $\mathcal{F}$ of subsets $U$ is called a $p$-family if for all $X \in \mathcal{F}$, $|X| = p$.

We call a function $f : 2^U \to \mathbb{N}$ *additive* if for any subsets $X$ and $Y$ of $U$ we have that $f(X) + f(Y) = f(X \cup Y) - f(X \cap Y)$.

A monomial $Z = x_1^{s_1} \cdots x_n^{s_n}$ of a polynomial $P(x_1, \ldots, x_n)$ is called *multilinear* if $s_i \in \{0, 1\}$ for all $i \in \{1, \ldots, n\}$. We say a monomial $Z = x_1^{s_1} \cdots x_n^{s_n}$ is *$k$-multilinear term* if $Z$ is multilinear and $\sum_{i=1}^{n} s_i = k$. Throughout the paper we use $\omega$ to denote the matrix multiplication exponent. The current best known bound on $\omega < 2.373$ [31].

### 2.1 Matroids and Representative Families

In this subsection we give definitions related to matroids and representative family. For a broader overview on matroids we refer to [26].

*Definition 2.1.* A pair $M = (E, \mathcal{I})$, where $E$ is a ground set and $\mathcal{I}$ is a family of subsets (called independent sets) of $E$, is a *matroid* if it satisfies the following conditions:

(I1) $\emptyset \in \mathcal{I}$.
(I2) If $A' \subseteq A$ and $A \in \mathcal{I}$ then $A' \in \mathcal{I}$.
(I3) If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there exists $e \in (B \setminus A)$ such that $A \cup \{e\} \in \mathcal{I}$.

The axiom (I2) is also called the hereditary property and a pair $(E, \mathcal{I})$ satisfying only (I2) is called hereditary family. An inclusion wise maximal set of $\mathcal{I}$ is called a *basis* of the matroid. Using axiom (I3) it is easy to show that all the bases of a matroid have the same size. This size is called the *rank* of the matroid $M$, and is denoted by $\text{rank}(M)$. The *uniform matroids* are among the simplest examples of matroids. A pair $M = (E, \mathcal{I})$ over an $n$-element ground set $E$, is called a uniform matroid if the family of independent sets is given by $\mathcal{I} = \{A \subseteq E \mid |A| \leq k\}$, where $k$ is some constant. This matroid is also denoted as $U_{n,k}$.

#### 2.1.1 *Linear Matroids and Representable Matroids.*

Let $A$ be a matrix over an arbitrary field $\mathbb{F}$ and let $E$ be the set of columns of $A$. Given $A$ we define the matroid $M = (E, \mathcal{I})$ as follows. A set $X \subseteq E$ is independent (that is $X \in \mathcal{I}$) if the corresponding columns are linearly independent over $\mathbb{F}$. The matroids that can be defined by such a construction are called *linear matroids*, and if a matroid can be defined by a matrix $A$ over a field $\mathbb{F}$, then we say that the matroid is representable over $\mathbb{F}$. That is, a matroid $M = (E, \mathcal{I})$ of rank $d$ is representable over a field $\mathbb{F}$ if there exist vectors in $\mathbb{F}^d$ correspond to the elements such that linearly independent sets of vectors correspond to independent sets of the matroid. A matroid $M = (E, \mathcal{I})$ is called *representable* or *linear* if it is representable over some field $\mathbb{F}$.

*2.1.2 Graphic Matroids.* Given a graph $G$, a graphic matroid $M = (E, \mathcal{I})$ is defined by taking elements as edges of $G$ (that is $E = E(G)$) and $F \subseteq E(G)$ is in $\mathcal{I}$ if it forms a spanning forest in the graph $G$. Consider the matrix $A_M$ with a row for each vertex $i \in V(G)$ and a column for each edge $e = ij \in E(G)$. In the column corresponding to $e = ij$, all entries are 0, except for a 1 in $i$ or $j$ (arbitrarily) and a $-1$ in the other. This is a representation over reals. To obtain a representation over a field $\mathbb{F}$, one needs to take the representation given above over reals and simply replace all $-1$ by the additive inverse of 1

Proposition 2.2 ([26]). *Graphic matroids are representable over any field of size at least* 2.

*2.1.3 Representative Family.* Now we define $q$-representative family of a given family and state Theorems [10] regarding its compuation.

*Definition 2.3 ($q$-**Representative Family** [10]).* Given a matroid $M = (E, \mathcal{I})$ and a family $\mathcal{S}$ of subsets of $E$, we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is *$q$-representative* for $\mathcal{S}$ if the following holds: for every set $Y \subseteq E$ of size at most $q$, if there is a set $X \in \mathcal{S}$ disjoint from $Y$ with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from $Y$ with $\widehat{X} \cup Y \in \mathcal{I}$. If $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is $q$-representative for $\mathcal{S}$ we write $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$.

In other words, if some independent set in $\mathcal{S}$ can be extended to a larger independent set by $q$ new elements, then there is a set in $\widehat{\mathcal{S}}$ that can be extended by the same $q$ elements. A weighted variant of $q$-representative families is defined as follows. It is useful for solving problems where we are looking for objects of maximum or minimum weight.

*Definition 2.4 (**Min/Max $q$-Representative Family** [10]).* Given a matroid $M = (E, \mathcal{I})$, a family $\mathcal{S}$ of subsets of $E$ and a non-negative weight function $w : \mathcal{S} \to \mathbb{N}$, we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is *min $q$-representative* (*max $q$-representative*) for $\mathcal{S}$ if the following holds: for every set $Y \subseteq E$ of size at most $q$, if there is a set $X \in \mathcal{S}$ disjoint from $Y$ with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from $Y$ with

(1) $\widehat{X} \cup Y \in \mathcal{I}$, and
(2) $w(\widehat{X}) \leq w(X)$ ($w(\widehat{X}) \geq w(X)$).

We use $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S}$) to denote a min $q$-representative (max $q$-representative) family for $\mathcal{S}$.

*Definition 2.5.* Given two families of independent sets $\mathcal{L}_1$ and $\mathcal{L}_2$ of a matroid $M = (E, \mathcal{I})$, we define

$$\mathcal{L}_1 \bullet \mathcal{L}_2 = \{X \cup Y \mid X \in \mathcal{L}_1 \wedge Y \in \mathcal{L}_2 \wedge X \cap Y = \emptyset \wedge X \cup Y \in \mathcal{I}\}.$$

For normal set families $\mathcal{A}$ and $\mathcal{B}$ (in uniform matroid of rank at least $\max_{A \in \mathcal{A}, B \in \mathcal{B}}(|A| + |B|)$), note that $\mathcal{A} \circ \mathcal{B} = \mathcal{A} \bullet \mathcal{B} = \{X \cup Y \mid X \in \mathcal{A} \wedge Y \in \mathcal{B} \wedge X \cap Y = \emptyset\}$.

We say that a family $\mathcal{S} = \{S_1, \ldots, S_t\}$ of independent sets is a *$p$-family* if each set in $\mathcal{S}$ is of size $p$. We state three lemmata providing basic results about representative families. These lemmata work for the weighted variant of representative families.

Lemma 2.6 ([10]). *Let $M = (E, \mathcal{I})$ be a matroid and $\mathcal{S}$ be a family of subsets of E. If $\mathcal{S}' \subseteq^q_{rep} \mathcal{S}$ and $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}'$, then $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$.*

Lemma 2.7 ([10]). *Let $M = (E, \mathcal{I})$ be a matroid and $\mathcal{S}$ be a family of subsets of E. If $\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_\ell$ and $\widehat{\mathcal{S}}_i \subseteq^q_{rep} \mathcal{S}_i$, then $\cup_{i=1}^{\ell} \widehat{\mathcal{S}}_i \subseteq^q_{rep} \mathcal{S}$.*

Lemma 2.8 ([10]). *Let $M = (E, \mathcal{I})$ be a matroid of rank $k$, $\mathcal{S}_1$ be a $p_1$-family of independent sets, and $\mathcal{S}_2$ be a $p_2$-family of independent sets such that $\widehat{\mathcal{S}}_1 \subseteq^{k-p_1}_{rep} \mathcal{S}_1$ and $\widehat{\mathcal{S}}_2 \subseteq^{k-p_2}_{rep} \mathcal{S}_2$. Then $\widehat{\mathcal{S}}_1 \bullet \widehat{\mathcal{S}}_2 \subseteq^{k-p_1-p_2}_{rep} \mathcal{S}_1 \bullet \mathcal{S}_2$.*

Theorem 2.9 ([10]). *Let $M = (E, \mathcal{I})$ be a linear matroid of rank $p + q = k$, $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a $p$-family of independent sets and $w : \mathcal{S} \to \mathbb{N}$ be a non-negative weight function. Then there exists $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S}$) of size $\binom{p+q}{p}$. Moreover, given a representation $A_M$ of $M$ over a field $\mathbb{F}$, we can find $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^q_{maxrep} \mathcal{S}$) of size at most $\binom{p+q}{p}$ in $O\left(\binom{p+q}{p} tp^{\omega} + t\binom{p+q}{q}^{\omega-1}\right)$ operations over $\mathbb{F}$.*

It is shown in [21] that a theorem similar to Theorem 2.9 can be obtained even when the rank of the input matroid is not bounded, through a deterministic truncation of linear matroids. For uniform matroids faster algorithms are known.

Theorem 2.10 ([10]). *There is an algorithm that given a $p$-family $\mathcal{A}$ of sets over a universe $U$ of size $n$, an integer $q$, and a non-negative weight function $w : \mathcal{A} \to \mathbb{N}$ with maximum value at most $W$, computes in time $O(|\mathcal{A}| \cdot \log |\mathcal{A}| \cdot \log W + |\mathcal{A}| \cdot \left(\frac{p+q}{q}\right)^q \cdot 2^{o(p+q)} \cdot \log n)$ a subfamily $\widehat{\mathcal{A}} \subseteq \mathcal{A}$ such that $|\widehat{\mathcal{A}}| \le \binom{p+q}{p} \cdot 2^{o(p+q)}$ and $\widehat{\mathcal{A}} \subseteq^q_{minrep} \mathcal{A}$ ($\widehat{\mathcal{A}} \subseteq^q_{maxrep} \mathcal{A}$).*

## 3 REPRESENTATIVE FAMILY COMPUTATION FOR PRODUCT FAMILIES

In this section we design a faster algorithm to find $q$-representative family for product families. Our algorithm for $q$-representative family for product families relies on the construction of *n-p-q-separating collection* defined in [10]. We start with the formal definition of *n-p-q-separating collection.*

*Definition 3.1.* An $n$-$p$-$q$-separating collection $C$ is a tuple $(\mathcal{F}, \chi, \chi')$, where $\mathcal{F}$ is a family of sets over a universe $U$ of size $n$, $\chi$ is a function from $\binom{U}{\le p}$ to $2^{\mathcal{F}}$ and $\chi'$ is a function from $\binom{U}{\le q}$ to $2^{\mathcal{F}}$ such that the following properties are satisfied

(1) for every $A \in \binom{U}{\le p}$ and $F \in \chi(A)$, $A \subseteq F$;
(2) for every $B \in \binom{U}{\le q}$ and $F \in \chi'(B)$, $F \cap B = \emptyset$;
(3) for every pairwise disjoint sets $A_1 \in \binom{U}{p_1}, A_2 \in \binom{U}{p_2}, \cdots, A_r \in \binom{U}{p_r}$ and $B \in \binom{U}{q}$ such that $p_1 + \cdots + p_r = p$, $\exists F \in \chi(A_1) \cap \chi(A_2) \ldots \chi(A_r) \cap \chi'(B)$.

The size of $(\mathcal{F}, \chi, \chi')$ is $|\mathcal{F}|$, the $(\chi, p')$-degree of $(\mathcal{F}, \chi, \chi')$ for $p' \le p$ is

$$\max_{A \in \binom{U}{p'}} |\chi(A)|,$$

and the $(\chi', q')$-degree of $(\mathcal{F}, \chi, \chi')$ for $q' \le q$ is

$$\max_{B \in \binom{U}{q'}} |\chi'(B)|.$$

A *construction* of separating collections is a data structure, that given $n$, $p$ and $q$ initializes and outputs a family $\mathcal{F}$ of sets over the universe $U$ of size $n$. After the initialization one can query the data structure by giving it a set $A \in \binom{U}{\le p}$ or $B \in \binom{U}{\le q}$, and the data structure then outputs a family $\chi(A) \subseteq 2^{\mathcal{F}}$ or $\chi'(B) \subseteq 2^{\mathcal{F}}$, respectively. Together the tuple $C = (\mathcal{F}, \chi, \chi')$ computed by the data structure should form an *n-p-q-separating collection.*

Lemma 3.2 ([10]). *Given $0 < x < 1$, there is a construction of an n-p-q- separating collection with the following parameters*

- *size,* $\zeta(n,p,q) \le 2^{O(\frac{p+q}{\log\log(p+q)})} \cdot \frac{1}{x^p(1-x)^q} \cdot (p+q)^{O(1)} \cdot \log n$
- *initialization time,* $\tau_I(n,p,q) \le 2^{O(\frac{p+q}{\log\log(p+q)})} \cdot \frac{1}{x^p(1-x)^q} \cdot (p+q)^{O(1)} \cdot n\log n$
- $(\chi, p')$*-degree,* $\Delta_{(\chi,p')}(n,p,q) \le 2^{O(\frac{p+q}{\log\log(p+q)})} \cdot \frac{1}{x^{p-p'}(1-x)^q} \cdot (p+q)^{O(1)} \cdot \log n$
- $(\chi, p')$*-query time,* $Q_{(\chi,p')}(n,p,q) \le 2^{O(\frac{p+q}{\log\log(p+q)})} \cdot \frac{1}{x^{p-p'}(1-x)^q} \cdot (p+q)^{O(1)} \cdot \log n$
- $(\chi', q')$*-degree,* $\Delta_{(\chi',q')}(n,p,q) \le 2^{O(\frac{p+q}{\log\log(p+q)})} \cdot \frac{1}{x^p(1-x)^{q-q'}} \cdot (p+q)^{O(1)} \cdot \log n$
- $(\chi', q')$*-query time,* $Q_{(\chi',q')}(n,p,q) \le 2^{O(\frac{p+q}{\log\log(p+q)})} \cdot \frac{1}{x^p(1-x)^{q-q'}} \cdot (p+q)^{O(1)} \cdot \log n$

Let us provide first some intuition behind the algorithm computing a $q$-representative family for the product families. Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two families of sets over a universe $U$ of size $n$, where $\mathcal{L}_1$ is a $p_1$-family and $\mathcal{L}_2$ is a $p_2$-family. Any set in $\mathcal{L}_1 \circ \mathcal{L}_2$ is of the form $A \cup B$ where $A \in \mathcal{L}_1$, $B \in \mathcal{L}_2$, and $A \cap B = \emptyset$. Let $p = p_1 + p_2$ and $q = k - p$. We want to find a small subfamily $\widehat{\mathcal{L}}$ of $\mathcal{L}_1 \circ \mathcal{L}_2$ satisfying the following property: For every set $C$ of size at most $q$, if $(A \cup B) \cap C = \emptyset$, where $A \cup B \in \mathcal{L}_1 \circ \mathcal{L}_2$, then there are sets $A' \in \mathcal{L}_1$ and $B' \in \mathcal{L}_2$ such that $A' \cup B' \in \widehat{\mathcal{L}}$, $A' \cap B' = \emptyset$, and $(A' \cup B') \cap C = \emptyset$. To construct such a subfamily, we build two separating collections. The first $n$-$p$-$q$-separating collection $(\mathcal{F}, \chi_{\mathcal{F}}, \chi'_{\mathcal{F}})$ is used to take care of the disjointness between $A \cup B$ and $C$. The second $n$-$p_1$-$p_2$-separating collection $(\mathcal{H}, \chi_{\mathcal{H}}, \chi'_{\mathcal{H}})$ is for taking care of the disjointness between the sets in $\mathcal{L}_1$ and $\mathcal{L}_2$ (i.e, between $A$ and $B$). For any tuple $(A, B, C)$ of sets, where $A \in \mathcal{L}_1$, $B \in \mathcal{L}_2$, $A \cap B = \emptyset$, and $C$ is a set of size at most $q$ such that $(A \cup B) \cap C = \emptyset$, there is a pair of of sets $F \in \mathcal{F}$ and $H \in \mathcal{H}$ with the following property: $A \subseteq H$, $B \cap H = \emptyset$, $A \cup B \subseteq F$, and $C \cap F = \emptyset$. Hence to keep the $q$-representative family, it is sufficient to keep for every pair of sets $F$ and $H$ only one set $A' \cup B' \in \widehat{\mathcal{L}}$, where $A' \subseteq H$, $B' \cap H = \emptyset$, and $A' \cup B' \subseteq F$.

We are ready to give the main theorem about product families using the constructions of *n-p-q-*separating collections.

Theorem 3.3. *Let $\mathcal{L}_1$ be a $p_1$-family of sets and $\mathcal{L}_2$ be a $p_2$-family of sets over a universe $U$ of size $n$. Let $w : 2^U \to \mathbb{N}$ be an additive weight function. Let $\mathcal{L} = \mathcal{L}_1 \circ \mathcal{L}_2$ and $p = p_1 + p_2$. For any $0 < x_1, x_2 < 1$, there exist $\widehat{\mathcal{L}} \subseteq^{k-p_1-p_2}_{minrep} \mathcal{L}$ of size $x_1^{-p}(1-x_1)^{-(k-p)} \cdot 2^{o(k)} \cdot \log n$ and it can be computed in time*

$$O\left(\frac{z(n,k,W)}{x_1^p(1-x_1)^q} + \frac{z(n,k,W)}{x_2^{p_1}(1-x_2)^{p_2}} + \frac{|\mathcal{L}_1| \cdot z(n,k,W)}{x_1^{p_2}(1-x_1)^q(1-x_2)^{p_2}} + \frac{|\mathcal{L}_2| \cdot z(n,k,W)}{x_1^{p_1}(1-x_1)^q x_2^{p_1}}\right),$$

*where $z(n,k,W) = 2^{o(k)} n \log n \cdot \log W$ and $W$ is the maximum weight defined by $w$.*

Proof. We set $p = p_1 + p_2$ and $q = k - p$. To obtain the desired construction we first define an auxiliary graph and then use it to obtain the $q$-representative for the product family $\mathcal{L}$. We first obtain two families of separating collections.

- Apply Lemma 3.2 for $0 < x_1 < 1$ and construct an $n$-$p$-$q$-separating collection $(\mathcal{F}, \chi_{\mathcal{F}}, \chi'_{\mathcal{F}})$ of size $2^{O(\frac{p+q}{\log\log(p+q)})} \cdot \frac{1}{x_1^p(1-x_1)^q} \cdot (p+q)^{O(1)} \log n$ in time linear in the size of $\mathcal{F}$.

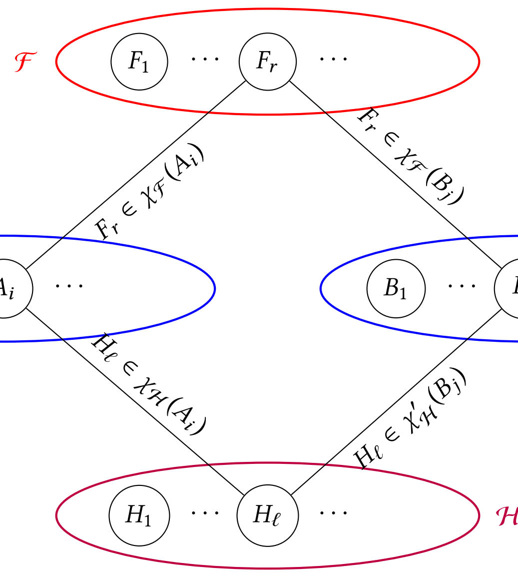


Fig. 1. Graph constructed from $\mathcal{L}_1, \mathcal{L}_2, \mathcal{F}$ and $\mathcal{H}$

- Apply Lemma 3.2 for $0 < x_2 < 1$ and construct an $n$-$p_1$-$p_2$-separating collection $(\mathcal{H}, \chi_{\mathcal{H}}, \chi'_{\mathcal{H}})$ of size $2^{O(\frac{p_1+p_2}{\log\log(p_1+p_2)})} \cdot \frac{1}{x_2^{p_1}(1-x_2)^{p_2}} \cdot (p_1+p_2)^{O(1)} \log n$ in time linear in the size of $\mathcal{H}$.

Now we construct a graph $G = (V, E)$ where the vertex set $V$ contains a vertex each for sets in $\mathcal{F} \uplus \mathcal{H} \uplus \mathcal{L}_1 \uplus \mathcal{L}_2$. For clarity of presentation we name the vertices by the corresponding set. Thus, the vertex set $V = \mathcal{F} \uplus \mathcal{H} \uplus \mathcal{L}_1 \uplus \mathcal{L}_2$. The edge set $E = E_1 \uplus E_2 \uplus E_3 \uplus E_4$, where each $E_i$ for $i \in \{1, 2, 3, 4\}$ is defined as follows (see Figure 1).

$$\begin{aligned}
E_1 &= \Big\{(A, F) \,\Big|\, A \in \mathcal{L}_1,\ F \in \chi_{\mathcal{F}}(A)\Big\}\\
E_2 &= \Big\{(B, F) \,\Big|\, B \in \mathcal{L}_2,\ F \in \chi_{\mathcal{F}}(B)\Big\}\\
E_3 &= \Big\{(A, H) \,\Big|\, A \in \mathcal{L}_1,\ H \in \chi_{\mathcal{H}}(A)\Big\}\\
E_4 &= \Big\{(B, F) \,\Big|\, B \in \mathcal{L}_2,\ F \in \chi'_{\mathcal{H}}(B)\Big\}
\end{aligned}$$

Thus $G$ is essentially a 4-partite graph.

*Algorithm.* The construction of $\widehat{\mathcal{L}}$ is as follows. For a set $F \in \mathcal{F}$, we call a pair of sets $(A, B)$ *cyclic*, if $A \in \mathcal{L}_1$, $B \in \mathcal{L}_2$ and there exists $H \in \mathcal{H}$ such that $FAHB$ forms a cycle of length four in $G$. Let $\mathcal{J}(F)$ denote the family of cyclic pairs for a set $F \in \mathcal{F}$ and

$$w_F = \min_{(A,B) \in \mathcal{J}(F)} w(A) + w(B).$$

We obtain the family $\widehat{\mathcal{L}}$ by adding $A \cup B$ for every set $F \in \mathcal{F}$ such that $(A, B) \in \mathcal{J}(F)$ and $w(A) + w(B) = w_F$. Indeed, if the family $\mathcal{J}(F)$ is empty then we do not add any set to $\widehat{\mathcal{L}}$ corresponding to $F$. The procedure to find the smallest weight $A \cup B$ for any $F$ is as follows. We first mark the vertices of $N_G(F)$ (the neighbors of $F$). Now we mark the neighbors of $\mathcal{P} = (N_G(F) \cap \mathcal{L}_1)$ in $\mathcal{H}$. For every marked vertex $H \in \mathcal{H}$, we associate a set $A$ of minimum weight such that $A \in (\mathcal{P} \cap N_G(H))$. This can be done sequentially as follows. Let $\mathcal{P} = \{S_1, \ldots, S_\ell\}$. Now iteratively visit the neighbors of $S_i$ in $\mathcal{H}$, $i \in [\ell]$, and for each vertex of $\mathcal{H}$ store the smallest weight vertex $S \in \mathcal{P}$ it has seen so far. After this we have a marked set of vertices in $\mathcal{H}$ such that with each marked vertex $H$ in $\mathcal{H}$ we stored a smallest weight marked vertex in $\mathcal{L}_1$ which is a neighbor of $H$. Now for each marked vertex $B$ in $\mathcal{L}_2$, we go through the neighbors of $B$ in the marked set of vertices in $\mathcal{H}$ and associate (if possible) a second vertex (which is a minimum weighted marked neighbor from $\mathcal{L}_2$) with each marked vertex in $\mathcal{H}$. We obtain a pair of sets $(A, B) \in \mathcal{J}(F)$ such that $w(A) + w(B) = w_F$. This can be easily done by keeping a variable that stores a minimum weighted $A \cup B$ seen after every step of marking procedure. Since for each $F \in \mathcal{F}$ we add at most one set to $\widehat{\mathcal{L}}$, the size of $\widehat{\mathcal{L}}$ follows.

*Correctness.* We first show that $\widehat{\mathcal{L}} \subseteq \mathcal{L}$. Towards this we only need to show that for every $A \cup B \in \widehat{\mathcal{L}}$ we have that $A \cap B = \emptyset$. Observe that if $A \cup B \in \widehat{\mathcal{L}}$ then there exist $F \in \mathcal{F}$ and $H \in \mathcal{H}$ such that $FAHB$ forms a cycle of length four in the graph $G$. So $H \in \chi_{\mathcal{H}}(A)$ and $H \in \chi'_{\mathcal{H}}(B)$. This means $A \subseteq H$ and $B \cap H = \emptyset$. So we conclude $A$ and $B$ are disjoint and hence $\widehat{\mathcal{L}} \subseteq \mathcal{L}$. We also need to show that if there exist pairwise disjoint sets $A \in \mathcal{L}_1, B \in \mathcal{L}_2, C \in \binom{U}{q}$, then there exist $\widehat{A} \in \mathcal{L}_1, \widehat{B} \in \mathcal{L}_2$ such that $\widehat{A} \cup \widehat{B} \in \widehat{\mathcal{L}}$, $\widehat{A}, \widehat{B}, C$ are pairwise disjoint and $w(\widehat{A}) + w(\widehat{B}) \leq w(A) + w(B)$. By the property of separating collections $(\mathcal{F}, \chi_{\mathcal{F}}, \chi'_{\mathcal{F}})$ and $(\mathcal{H}, \chi_{\mathcal{H}}, \chi'_{\mathcal{H}})$, we know that there exists $F \in \chi_{\mathcal{F}}(A) \cap \chi_{\mathcal{F}}(B) \cap \chi'_{\mathcal{F}}(C)$, $H \in \chi_{\mathcal{H}}(A) \cap \chi'_{H}(B)$. This implies that $FAHB$ forms a cycle of length four in the graph $G$. Hence in the construction of $\widehat{\mathcal{L}}$, we should have chosen $\widehat{A} \in \mathcal{L}_1$ and $\widehat{B} \in \mathcal{L}_2$ corresponding to $F$ such that $w(\widehat{A}) + w(\widehat{B}) \leq w(A) + w(B)$ and added to $\widehat{\mathcal{L}}$. So we know that $F \in \chi_{\mathcal{F}}(\widehat{A}) \cap \chi_{\mathcal{F}}(\widehat{B})$. Now we claim that $\widehat{A}, \widehat{B}$ and $C$ are pairwise disjoint. Since $\widehat{A} \cup \widehat{B} \in \widehat{\mathcal{L}}$, $\widehat{A} \cap \widehat{B} = \emptyset$. Finally, since $F \in \chi_{\mathcal{F}}(\widehat{A}) \cap \chi_{\mathcal{F}}(\widehat{B})$ and $F \in \chi'_{\mathcal{F}}(C)$, we get $\widehat{A}, \widehat{B} \subseteq F$ and $F \cap C = \emptyset$ which implies $C$ is disjoint from $\widehat{A}$ and $\widehat{B}$. This completes the correctness proof.

*Running Time Analysis.* We first consider the time $T_G$ to construct the graph $G$. We can construct $\mathcal{F}$ in time $2^{O(\frac{k}{\log\log k})} \cdot \frac{1}{x_1^p(1-x_1)^q} \cdot (p+q)^{O(1)} \cdot n \log n$. We can construct $\mathcal{H}$ in time $2^{O(\frac{p}{\log\log p})} \cdot \frac{1}{x_2^{p_1}(1-x_2)^{p_2}} \cdot (p_1+p_2)^{O(1)} \cdot n \log n$. Now to add edges in the graph we do as follows. For each vertex in $\mathcal{L}_1 \cup \mathcal{L}_2$, we query the data structure created, spending the query time mentioned in Lemma 3.2, and add edges to the vertices in $\mathcal{F} \cup \mathcal{H}$ from it. So the running time to construct $G$ is

$$\begin{aligned} T_G \quad \leq \quad & 2^{O(\frac{k}{\log\log(k)})} k^{O(1)} n \log n \Big( \frac{1}{x_1^p (1-x_1)^q} + \frac{1}{x_2^{p_1}(1-x_2)^{p_2}} + \frac{|\mathcal{L}_1|}{x_1^{p_2}(1-x_1)^q} \\ & + \frac{|\mathcal{L}_2|}{x_1^{p_1}(1-x_1)^q} + \frac{|\mathcal{L}_1|}{(1-x_2)^{p_2}} + \frac{|\mathcal{L}_2|}{x_2^{p_1}} \Big). \end{aligned}$$

Now we bound the time $T_C$ taken to construct $\widehat{\mathcal{L}}$ from $G$. To do the analysis we see how may times a vertex $A$ in $\mathcal{L}_1 \cup \mathcal{L}_2$ is visited. It is exactly equal to the product of the degree of $A$ to $\mathcal{F}$ (denoted by $\text{degree}_{\mathcal{F}}(A)$) and the degree of $A$ to $\mathcal{H}$ (denoted by $\text{degree}_{\mathcal{H}}(A)$). Also note that two weights can be compared in $O(\log W)$ time. Then

$$
\begin{aligned}
T_C &\leq \log W\left(\sum_{A\in\mathcal{L}_1} \text{degree}_{\mathcal{F}}(A)\cdot\text{degree}_{\mathcal{H}}(A) + \sum_{A\in\mathcal{L}_2} \text{degree}_{\mathcal{F}}(A)\cdot\text{degree}_{\mathcal{H}}(A)\right)\\
&\leq \log W\Bigg(\sum_{A\in\mathcal{L}_1} \Delta_{(\chi_{\mathcal{F}},p_1)}(n,p,q)\cdot\Delta_{(\chi_{\mathcal{H}},p_1)}(n,p_1,p_2) + \\
&\qquad\qquad \sum_{A\in\mathcal{L}_2} \Delta_{(\chi_{\mathcal{F}},p_2)}(n,p,q)\cdot\Delta_{(\chi'_{\mathcal{H}},p_2)}(n,p_1,p_2)\Bigg)\\
&\leq 2^{O(\frac{k}{\log\log(k)})}k^{O(1)}\log^2 n\log W\left(\frac{|\mathcal{L}_1|}{x_1^{p_2}(1-x_1)^q(1-x_2)^{p_2}} + \frac{|\mathcal{L}_2|}{x_1^{p_1}(1-x_1)^q x_2^{p_1}}\right).
\end{aligned}
$$

So the total running time $T$ is,

$$
\begin{aligned}
T &= T_G + T_C\\
&\leq 2^{O(\frac{k}{\log\log(k)})}k^{O(1)} n\log n\cdot\log W\Bigg(\frac{1}{x_1^p(1-x_1)^q} + \frac{1}{x_2^{p_1}(1-x_2)^{p_2}}\\
&\qquad + \frac{|\mathcal{L}_1|}{x_1^{p_2}(1-x_1)^q(1-x_2)^{p_2}} + \frac{|\mathcal{L}_2|}{x_1^{p_1}(1-x_1)^q x_2^{p_1}}\Bigg).
\end{aligned}
$$

This completes the proof of the theorem. □

Now we give a ready to use corollary for Theorem 3.3.

COROLLARY 3.4. *Let $\mathcal{L}_1$ be a $p_1$-family of sets and $\mathcal{L}_2$ be a $p_2$-family of sets over a universe $U$ of size $n$. Furthermore, let $w : 2^U \to \mathbb{N}$ be an additive weight function, $|\mathcal{L}_1| = \binom{k}{p_1}\cdot 2^{o(k)}$, $|\mathcal{L}_2| = \binom{k}{p_2}\cdot 2^{o(k)}$, $\mathcal{L} = \mathcal{L}_1 \circ \mathcal{L}_2$, $p = p_1 + p_2$ and $q = k - p$. There exists $\widehat{\mathcal{L}} \subseteq^q_{minrep} \mathcal{L}$ of size $\binom{k}{p}\cdot 2^{o(k)}$ and it can be computed in time*

$$
\min_{0<x_1,x_2<1} O\left(\frac{z(n,k,W)}{x_2^{p_1}(1-x_2)^{p_2}} + \frac{\binom{k}{p_1}\cdot z(n,k,W)}{x_1^{p_2}(1-x_1)^q(1-x_2)^{p_2}} + \frac{\binom{k}{p_2}\cdot z(n,k,W)}{x_1^{p_1}(1-x_1)^q x_2^{p_1}} + \frac{(\frac{k}{q})^q\cdot z(n,k,W)}{x_1^p(1-x_1)^q}\right).
$$

*Here $z(n,k,W) = 2^{o(k)} n\log n\cdot\log W$ and $W$ is the maximum weight defined by $w$.*

PROOF. We apply Theorem 3.3 for $0 < x_1, x_2 < 1$ and find $\mathcal{L}' \subseteq^q_{minrep} \mathcal{L}$ of size $x_1^{-p}(1-x_1)^{-q}2^{o(k)}\cdot\log n$ in time $T_1 = O(\frac{z(n,k,W)}{x_1^p(1-x_1)^q} + \frac{z(n,k,W)}{x_2^{p_1}(1-x_2)^{p_2}} + \frac{z(n,k,W)\cdot|\mathcal{L}_1|}{x_1^{p_2}(1-x_1)^q(1-x_2)^{p_2}} + \frac{z(n,k,W)\cdot|\mathcal{L}_2|}{x_1^{p_1}(1-x_1)^q x_2^{p_1}})$. Now we apply Theorem 2.10 and get $\widehat{\mathcal{L}} \subseteq^q_{minrep} \mathcal{L}'$ of size $\binom{k}{p}\cdot 2^{o(k)}$ in time $T_2 = O\left(x_1^{-p}(1-x_1)^{-q}\left(\frac{k}{q}\right)^q 2^{o(k)}\cdot\log^2 n\cdot\log W\right)$. Due to Lemma 2.6, $\widehat{\mathcal{L}} \subseteq^q_{minrep} \mathcal{L}$. Now we choose $x_1, x_2$ such that $T_1 + T_2$ is minimized. So the total running time $T$ to construct $\widehat{\mathcal{L}}$ is

$$
\begin{aligned}
T &= \min_{x_1,x_2}(T_1 + T_2)\\
&= \min_{x_1,x_2} O\Bigg(\frac{z(n,k,W)}{x_2^{p_1}(1-x_2)^{p_2}} + \frac{z(n,k,W)\cdot|\binom{k}{p_1}|}{x_1^{p_2}(1-x_1)^q(1-x_2)^{p_2}} + \\
&\qquad \frac{z(n,k,W)\cdot|\binom{k}{p_2}|}{x_1^{p_1}(1-x_1)^q x_2^{p_1}} + \frac{z(n,k,W)\cdot(\frac{k}{q})^q}{x_1^p(1-x_1)^q}\Bigg).
\end{aligned}
$$

This completes the proof. □

## 4 REPRESENTATIVE FAMILY COMPUTATION FOR PRODUCT FAMILIES OF A LINEAR MATROID

In this section we give an algorithm to compute $q$-representative family for product families of a linear matroid. That is, given a matroid $M = (E, \mathcal{I})$, families of independent sets $\mathcal{A}$ and $\mathcal{B}$ of sets of sizes $p_1$ and $p_2$, respectively, and a positive integer $q$, we compute $\widehat{\mathcal{F}} \subseteq^q_{rep} \mathcal{F}$, where $\mathcal{F} = \mathcal{A} \bullet \mathcal{B}$, of size $\binom{p_1+p_2+q}{p_1+p_2}$ efficiently. We compute a $q$-representative family for $\mathcal{F}$ in two steps. In the first step we compute an intermediate $q$-representative family and then apply Theorem 2.9 to compute $q$-representative family of the desired size. The intermediate $q$-representative family is obtained by computing $q$-representative families of *slices*, $\mathcal{A} \bullet \{B\}$ for all $B \in \mathcal{B}$, and then taking its union. We start with the following lemma that will be central to our faster algorithm for computing the desired $q$-representative family for a product family of a linear matroid.

Lemma 4.1 (Slice Computation Lemma). *Let $M = (E, \mathcal{I})$ be a linear matroid of rank $k$, $\mathcal{L}$ be a $p_1$-family of independent sets of $M$ and $S \in \mathcal{I}$ of size $p_2$. Furthermore, let $w : \mathcal{L} \bullet \{S\} \rightarrow \mathbb{N}$ be a non-negative weight function. Then given a representation $A_M$ of $M$ over a field $\mathbb{F}$, we can find $\widehat{\mathcal{L} \bullet \{S\}} \subseteq^{k-p_1-p_2}_{minrep} \mathcal{L} \bullet \{S\}$ of size at most $\binom{k-p_2}{p_1}$ in $O\left(\binom{k-p_2}{p_1}|\mathcal{L}|p_1^{\omega} + |\mathcal{L}|\binom{k-p_2}{p_1}^{\omega-1}\right)$ operations over $\mathbb{F}$.*

Proof. Observe that $\mathcal{L} \bullet \{S\}$ is a $p_1 + p_2$-family of independent sets of $M$ and all sets in $\mathcal{L} \bullet \{S\}$ contain $S$ as a subset. Let $A_M$ the matrix representing the matroid $M$ over a field $\mathbb{F}$. Without loss of generality we can assume that the first $p_2$ columns of $A_M$ correspond to the elements in $S$. Furthermore, we can also assume that the first $p_2$ columns and $p_2$ rows form an identity matrix $I_{p_2 \times p_2}$. That is, if $S$ denotes the first $p_2$ columns and $Z$ denotes the first $p_2$ rows then the submatrix $A_M[Z, S]$ is $I_{p_2 \times p_2}$. The reason for the last assertion is that if the matrix is not in the required form then we can apply elementary row operations and obtain the matrix in the desired form. This also allows us to assume that the number of rows in $A_M$ is $k$. So $A_M$ have the following form.

$$\left(\begin{array}{c|c} I_{p_2 \times p_2} & A \\ \hline 0 & B \end{array}\right)$$

Let $A_{M/S}$ be the matrix obtained after deleting first $p_2$ rows and first $p_2$ columns from $A_M$. That is, $A_{M/S}$= $B$. Let $M/S = (E_s, \mathcal{I}_s)$ be the matriod represented by the matrix $A_{M/S}$ on the underlying ground set $E_s = E \setminus S$. Observe that rank($M/S$)=rank($B$)= $k - p_2$, else rank($A_M$) would become strictly smaller than $k$. Let $e_1, e_2, \ldots, e_{p_2}$ be the first $p_2$ column vectors of $A_M$, i.e., they are columns corresponding to the elements of $S$. For a column vector $v$ in $A_M$, $\bar{v}$ is used to denote the column vector restricted to the matrix $A_{M/S}$ (i.e., $\bar{v}$ contains the last $k - p_2$ entries of $v$).

Now consider the set $\mathcal{L}(S) = \{X \mid X \cup S \in \mathcal{L} \bullet \{S\}\}$. We also define a new non-negative weight function $w' : \mathcal{L}(S) \rightarrow \mathbb{N}$ as follows: $w'(X) = w(X \cup S)$. We would like to compute $k - p_2$ representative for $\mathcal{L}(S)$. Towards that goal we first show that $\mathcal{L}(S)$ is a $p_1$-family of independent sets of $M/S$. Let $X \in \mathcal{L}(S)$. We know that $X \cup S \in \mathcal{I}$. Let $v_1, v_2, \ldots, v_{p_1}$ be the column vectors in $A_M$ corresponding to the elements in $X$. Suppose $X \notin \mathcal{I}_s$. Then there exist coefficients $\lambda_1, \ldots, \lambda_{p_1}$

such that $\lambda_1\bar{v}_1 + \lambda_2\bar{v}_2 + \cdots + \lambda_{p_1}\bar{v}_{p_1} = \vec{0}$ and at least one of them is non-zero. Then

$$\lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_{p_1} v_{p_1} = \begin{pmatrix} a_1 \\ \vdots \\ a_{p_2} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

This implies that $-a_1e_1 - a_2e_2 - \cdots - a_{p_2}e_{p_2} + \lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_{p_1} v_{p_1} = \vec{0}$, which contradicts the fact that $S \cup X \in \mathcal{I}$. Hence $X \in \mathcal{I}_s$ and $\mathcal{L}(S)$ is a $p_1$-family of independent sets of $M/S$.

Now we apply Theorem 2.9 and find $\widehat{\mathcal{L}(S)} \subseteq^{k-p_1-p_2}_{minrep} \mathcal{L}(S)$ of size $\binom{k-p_2}{p_1}$, by considering $\mathcal{L}(S)$ as a $p_1$-family of independent sets of the matroid $M/S$. We claim that $\widehat{\mathcal{L}(S)} \bullet \{S\} \subseteq^{k-p_1-p_2}_{minrep} \mathcal{L} \bullet \{S\}$. Let $X \cup S \in \mathcal{L} \bullet \{S\}$ and $Y \subseteq E \setminus (X \cup S)$ such that $|Y| = k - p_1 - p_2$ and $X \cup S \cup Y \in \mathcal{I}$. We need to show that there exists a $\widehat{X} \in \widehat{\mathcal{L}(S)}$ such that $\widehat{X} \cup S \cup Y \in \mathcal{I}$ and $w(\widehat{X} \cup S) \leq w(X \cup S)$. We start by showing that that $X \cup Y \in \mathcal{I}_s$. Let $v_1, v_2, \ldots, v_{k-p_2}$ be the column vectors in $A_M$ corresponding to the elements of $X \cup Y$. Suppose $X \cup Y \notin \mathcal{I}_s$. Then there exist coefficients $\lambda_1, \ldots, \lambda_{k-p_2}$ such that $\lambda_1\bar{v}_1 + \lambda_2\bar{v}_2 + \cdots + \lambda_{k-p_2}\bar{v}_{k-p_2} = \vec{0}$ and at least one of them is non-zero. Then we have the following.

$$\lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_{k-p_2} v_{k-p_2} = \begin{pmatrix} b_1 \\ \vdots \\ b_{p_2} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

However, this implies that $-b_1e_1 - b_2e_2 - \cdots - b_{p_2}e_{p_2} + \lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_{k-p_2} v_{k-p_2} = \vec{0}$, which contradicts the fact that $S \cup X \cup Y \in \mathcal{I}$. Hence $X \cup Y \in \mathcal{I}_s$. Since $\widehat{\mathcal{L}(S)} \subseteq^{k-p_1-p_2}_{minrep} \mathcal{L}(S)$, there exists a set $\widehat{X} \in \mathcal{L}(S)$, with $w'(\widehat{X}) \leq w'(X)$ (i.e $w(\widehat{X} \cup S) \leq w(X \cup S)$) and $\widehat{X} \cup Y \in \mathcal{I}_s$. We claim that $\widehat{X} \cup S \cup Y \in \mathcal{I}$. Let $u_1, u_2, \ldots, u_{k-p_2}$ be the column vectors in $A_M$ corresponding to the elements of $\widehat{X} \cup Y$. Suppose $\widehat{X} \cup S \cup Y \notin \mathcal{I}$. Then there exist coefficients $\alpha_1, \ldots, \alpha_k$ such that $\alpha_1e_1 + \alpha_2e_2 + \cdots + \alpha_{p_2}e_{p_2} + \alpha_{p_2+1}u_1 + \cdots + \alpha_k u_{k-p_2} = \vec{0}$ and at least one of the coefficients is non-zero. We claim that at least one of the coefficients among $\{\alpha_{p_2+1}, \ldots, \alpha_k\}$ is non-zero. Suppose not, then $\alpha_1e_1 + \cdots + \alpha_{p_2}e_{p_2} = 0$ and at least one of the coefficients among $\{\alpha_1, \ldots, \alpha_{p_2}\}$ is non-zero. This contradicts the fact that $S \in \mathcal{I}$. Since $\alpha_1e_1 + \cdots + \alpha_{p_2}e_{p_2} + \alpha_{p_2+1}u_1 + \cdots + \alpha_k u_{k-p_2} = \vec{0}$, we have that $\alpha_{p_2+1}\bar{u}_1 + \cdots + \alpha_k\bar{u}_{k-p_2} = \vec{0}$, where $\bar{u_j}$ are restrictions of $u_j$ to the last $k - p_2$ entries. Also note that at least one of the coefficients among $\{\alpha_{p_2+1}, \ldots, \alpha_k\}$ is non-zero. This contradicts our assumption that $\widehat{X} \cup Y \in \mathcal{I}_s$. Thus we have shown that $\widehat{X} \cup Y \cup S \in \mathcal{I}$. The size of $\widehat{\mathcal{L}(S)} \bullet \{S\}$ is $\binom{k-p_2}{p_1}$ and it can be found in $\mathcal{O}\left(\binom{k-p_2}{p_1}|\mathcal{L}|p_1^{\omega} + |\mathcal{L}|\binom{k-p_2}{p_1}^{\omega-1}\right)$ operations over $\mathbb{F}$. □

Now we are ready to prove the main theorem of this section by using Lemma 4.1.

Theorem 4.2. *Let $M = (E, \mathcal{I})$ be a linear matroid of rank $k$, $\mathcal{L}_1$ be a $p_1$-family of independent sets of $M$ and $\mathcal{L}_2$ be a $p_2$-family of independent sets of $M$. Given a representation $A_M$ of $M$ over a field $\mathbb{F}$,* *we can find* $\widehat{\mathcal{L}_1 \bullet \mathcal{L}_2} \subseteq^{k-p_1-p_2}_{minrep} \mathcal{L}_1 \bullet \mathcal{L}_2$ *of size at most* $\binom{k}{p_1+p_2}$ *in*

$$\mathcal{O}\left(|\mathcal{L}_2||\mathcal{L}_1|\binom{k-p_2}{p_1}^{\omega-1}p_1^{\omega} + |\mathcal{L}_2|\binom{k-p_2}{p_1}\binom{k}{p_1+p_2}^{\omega-1}(p_1+p_2)^{\omega}\right)$$

*operations over* $\mathbb{F}$.

Proof. Let $\mathcal{L}_2 = \{S_1, S_2, \ldots, S_\ell\}$. Then we have

$$\mathcal{L}_1 \bullet \mathcal{L}_2 = \bigcup_{i=1}^{\ell} \mathcal{L}_1 \bullet \{S_i\}.$$

By Lemma 2.7,

$$\mathcal{L} = \bigcup_{i=1}^{\ell} \widehat{\mathcal{L}_1 \bullet \{S_i\}} \subseteq^{k-p_1-p_2}_{minrep} \mathcal{L}_1 \bullet \mathcal{L}_2.$$

Using Lemma 4.1, for all $1 \le i \le \ell$, we find $\widehat{\mathcal{L}_1 \bullet \{S_i\}} \subseteq^{k-p_1-p_2}_{minrep} \mathcal{L}_1 \bullet \{S_i\}$ of size $\binom{k-p_2}{p_1}$ in $\mathcal{O}\left(\binom{k-p_2}{p_1}|\mathcal{L}_1|p_1^{\omega} + |\mathcal{L}_1|\binom{k-p_2}{p_1}^{\omega-1}\right) = \mathcal{O}\left(|\mathcal{L}_1|\binom{k-p_2}{p_1}^{\omega-1}p_1^{\omega}\right)$ operations over $\mathbb{F}$. Now we have that $|\mathcal{L}| = |\bigcup_{i=1}^{\ell} \widehat{\mathcal{L}_1 \bullet \{S_i\}}| \le |\mathcal{L}_2|\binom{k-p_2}{p_1}$. Now we apply Theorem 2.9 and find $\widehat{\mathcal{L}} \subseteq^{k-p_1-p_2}_{minrep} \mathcal{L}$ of size $\binom{k}{p_1+p_2}$. The number of operations, denoted by $T_1$, over $\mathbb{F}$ to find $\widehat{\mathcal{L}}$ from $\mathcal{L}$ is

$$\begin{aligned} T_1 &= \mathcal{O}\left(\binom{k}{p_1+p_1}|\mathcal{L}_2|\binom{k-p_2}{p_1}(p_1+p_2)^{\omega} + |\mathcal{L}_2|\binom{k-p_2}{p_1}\binom{k}{p_1+p_2}^{\omega-1}\right) \\ &= \mathcal{O}\left(|\mathcal{L}_2|\binom{k-p_2}{p_1}\binom{k}{p_1+p_2}^{\omega-1}(p_1+p_2)^{\omega}\right). \end{aligned}$$

By Lemma 2.6, $\widehat{\mathcal{L}} \subseteq^{k-p_1-p_2}_{minrep} \mathcal{L}_1 \bullet \mathcal{L}_2$. The number of operations, denoted by $T$, over $\mathbb{F}$ to find $\widehat{\mathcal{L}}$ from $\mathcal{L}_1$ and $\mathcal{L}_2$ is

$$\begin{aligned} T &= |\mathcal{L}_2| \cdot \mathcal{O}\left(|\mathcal{L}_1|\binom{k-p_2}{p_1}^{\omega-1}p_1^{\omega}\right) + T_1 \\ &= \mathcal{O}\left(|\mathcal{L}_2||\mathcal{L}_1|\binom{k-p_2}{p_1}^{\omega-1}p_1^{\omega} + |\mathcal{L}_2|\binom{k-p_2}{p_1}\binom{k}{p_1+p_2}^{\omega-1}(p_1+p_2)^{\omega}\right). \end{aligned}$$

This completes the proof of the theorem. □

The following form of Theorem 4.2 will be directly useful in some applications as we prune the size of the partial solutions in every step of the dynamic programming algorithm.

Corollary 4.3. *Let* $M = (E, \mathcal{I})$ *be a linear matroid of rank* $k$, $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *be two families of independent sets of* $M$ *and the number of sets of size* $p$ *in* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *be at most* $\binom{k+c}{p}$. *Here,* $c$ *is a fixed constant. Let* $\mathcal{L}_{r,i}$ *be the set of independent sets of size exactly* $i$ *in* $\mathcal{L}_r$ *for* $r \in \{1,2\}$. *Then for all the pairs* $i, j \in [k]$, *we can find* $\widehat{\mathcal{L}_{1,i} \bullet \mathcal{L}_{2,j}} \subseteq^{k-i-j}_{minrep} \mathcal{L}_{1,i} \bullet \mathcal{L}_{2,j}$ *of size* $\binom{k}{i+j}$, *in total of* $\mathcal{O}\left(k^{\omega}(2^{\omega}+2)^k + k^{\omega}2^{k(\omega-1)}3^k\right)$ *operations over* $\mathbb{F}$.

Proof. By using Theorem 4.2 we can find $\widehat{\mathcal{L}_{1,i} \bullet \mathcal{L}_{2,j}} \subseteq^{k-i-j}_{minrep} \mathcal{L}_{1,i} \bullet \mathcal{L}_{2,j}$ of size $\binom{k}{i+j}$ for any $i, j \in [k]$ in $\mathcal{O}\left(\binom{k+c}{j}\binom{k+c}{i}\binom{k-j}{i}^{\omega-1}i^{\omega} + \binom{k+c}{j}\binom{k-j}{i}\binom{k}{i+j}^{\omega-1}(i+j)^{\omega}\right)$ operations over $\mathbb{F}$. Let

$k' = k + c$. So the total number of operations, denoted by $T$, over $\mathbb{F}$ to find $\widehat{\mathcal{L}_{1,i} \bullet \mathcal{L}_{2,j}}$ for all $i, j \in [k]$ is,

$$
\begin{aligned}
T &= O\left(\left(\sum_{i=0}^{k}\sum_{j=0}^{k}\binom{k'}{j}\binom{k'}{i}\binom{k-j}{i}^{\omega-1} i^{\omega}\right) + \left(\sum_{i=0}^{k}\sum_{j=0}^{k}\binom{k'}{j}\binom{k-j}{i}\binom{k}{i+j}^{\omega-1}(i+j)^{\omega}\right)\right) \\
&= O\left(\left(k^{\omega}\sum_{i=0}^{k}\binom{k'}{i}\sum_{j=0}^{k}\binom{k'}{j}2^{(k-j)(w-1)}\right) + \left(k^{\omega}\sum_{j=0}^{k}\binom{k'}{j}\sum_{i=0}^{k-j}\binom{k-j}{i}\binom{k}{i+j}^{\omega-1}\right)\right) \\
&= O\left(\left(k^{\omega}2^{k(\omega-1)}\sum_{i=0}^{k}\binom{k'}{i}\left(1+\frac{1}{2^{(\omega-1)}}\right)^{k'}\right) + \left(k^{\omega}2^{k(w-1)}\sum_{j=0}^{k}\binom{k'}{j}\sum_{i=0}^{k-j}\binom{k-j}{i}\right)\right) \\
&= O\left(\left(k^{\omega}2^{k'}\left(2^{(\omega-1)}+1\right)^{k}\right) + \left(k^{\omega}2^{k(w-1)}\sum_{j=0}^{k}\binom{k'}{j}2^{k-j}\right)\right) \\
&= O\left(k^{\omega}2^{k}\left(2^{(\omega-1)}+1\right)^{k} + k^{\omega}2^{k(\omega-1)}3^{k}\right) \\
&= O\left(k^{\omega}\left(2^{\omega}+2\right)^{k} + k^{\omega}2^{k(\omega-1)}3^{k}\right).
\end{aligned}
$$

The above simplification completes the proof. □

## 5 APPLICATION I: MULTILINEAR MONOMIAL TESTING

In this section we first design a faster algorithm for a weighted version of $k$-MLD and then give an algorithm for an extension of this to a matroidal version. In the weighted version of $k$-MLD in addition to an arithmetic circuit $C$ over variables $X = \{x_1, x_2, \ldots, x_n\}$ representing a polynomial $P(X)$ over $\mathbb{Z}^+$, we are given an additive weight function $w : 2^X \to \mathbb{N}$. The task is that if there exists a $k$-multilinear term then find one with minimum weight. We call the weighted variant by $k$-wMLD. We start with the definition of an arithmetic circuit.

*Definition 5.1.* An *arithmetic circuit* $C$ over a commutative ring $R$ is a simple labelled directed acyclic graph with its internal nodes labeled by $+$ or $\times$ and leaves (in-degree zero nodes) labeled from $X \cup R$, where $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables. There is a node of out-degree zero, called the root node or the output gate. The size of $C$, $s(C)$, is the number of vertices in the graph.

It is well known that we can replace any arithmetic circuit $C$ with an equivalent circuit with fan-in two for all the internal nodes with quadratic blow up in the size. For an example, by replacing each node of in-degree greater than 2, with at most $s(C)$ many nodes of the same label and in-degree 2, we can convert a circuit $C$ to a circuit $C'$ of size $s(C') = s(C)^2$. So from now onwards we always assume that we are given a circuit of this form. We assume $W$ is the maximum weight defined by $w$.

THEOREM 5.2. $k$-wMLD *can be solved in time* $O(3.8408^k 2^{o(k)} s(C) n \log n \cdot \log W)$.

PROOF. An arithmetic circuit $C$ over $\mathbb{Z}^+$ with all leaves labelled from $X \cup \mathbb{Z}^+$ will represent sum of monomials with positive integer coefficients. With each multilinear term $\Pi_{j=1}^{\ell} x_{i_j}$ we associate a set $\{x_{i_1}, \ldots, x_{i_l}\} \subseteq X$. With any polynomial we can associate a family of subsets of $X$ which corresponds to the set of multilinear terms in it. Since $C$ is a directed acyclic graph, there exists a topological ordering $\pi = v_1, \ldots, v_n$, such that all the nodes corresponding to variables appear before any other gate and for every directed arc $uv$ we have that $u <_{\pi} v$. For a node $v_i$ of the circuit let $P_i(X)$ be the multivariate polynomial represented by the subcircuit containing all the nodes $w$ such that $w \leq_\pi v_i$. At every node we keep a family $\mathcal{F}_{v_i}^j$ of *j-multilinear term*, where $j \in \{1, \dots, k\}$. Let $\mathcal{F}_{v_i} = \bigcup_{x=1}^k \mathcal{F}_{v_i}^x$. Given a circuit $C$, if we compute associated family of subsets of $X$ for each node we can answer the question of having a $k$-multilinear term of minimum weight in the polynomial computed by $C$. But the size of the family of subsets could be exponential in $n$, the number of variables. That is, the size of $\mathcal{F}_{v_i}^j$ could be $\binom{n}{j}$. So instead of storing all subsets, we store a representative family for the associated family of subsets of each node. That is, we store $\widehat{\mathcal{F}_{v_i}^j} \subseteq_{minrep}^{k-j} \mathcal{F}_{v_i}^j$. The correctness of this step follows from the definition of $k-j$-representative family.

We make a dynamic programming algorithm to detect a multilinear monomial of order $k$ as follows. Our algorithm goes from left to right following the ordering given by $\pi$ and computes $\mathcal{F}_{v_i}$ from the families previously computed. The algorithm computes an appropriate representative family corresponding to each node of $C$. We show that we can compute a representative family $\mathcal{F}_v$ associated with any node $v$, where the number of subsets with $p$ elements in $\mathcal{F}_v$ is at most $\binom{k}{p}2^{o(k)}$. When $v$ is an input node then the associated family contains only one set. That is, if $v$ is labelled with $x_i$ then $\mathcal{F}_v = \{\{x_i\}\}$ and if $v$ is labelled from $\mathbb{Z}^+$ then $\mathcal{F}_v = \{\emptyset\}$. When $v$ is not an input node, then we have two cases.

**Addition Gate:** $v = v_1 + v_2$**.** Due to the left to right computation in the topological order, we have representative families $\mathcal{F}_{v_1}$ and $\mathcal{F}_{v_2}$ for $v_1$ and $v_2$, respectively, where the number of subsets with $p$ elements in $\mathcal{F}_{v_1}$ as well as in $\mathcal{F}_{v_2}$ will be at most $\binom{k}{p}2^{o(k)}$. The representative family corresponding to $v$ will be the representative family of $\mathcal{F}_{v_1} \cup \mathcal{F}_{v_2}$. We partition $\mathcal{F}_{v_1} \cup \mathcal{F}_{v_2}$ based on the size of subsets in it. Let $\mathcal{F}_{v_1} \cup \mathcal{F}_{v_2} = \biguplus_{p \leq k} \mathcal{H}_p$, where $\mathcal{H}_p$ contains all subsets of size $p$ in $\mathcal{F}_{v_1} \cup \mathcal{F}_{v_2}$. Note that $|\mathcal{H}_p| \leq 2\binom{k}{p}2^{o(k)}$. Now using Theorem 2.10, we can compute all $\widehat{\mathcal{H}_p} \subseteq_{minrep}^{k-p} \mathcal{H}_p$ in time

$$O\left(2^{o(k)} \log n \cdot \log W \cdot \sum_{p<k} \left\{ 2\binom{k}{p} \cdot \left(\frac{k}{k-p}\right)^{k-p} \right\}\right)$$

where $W$ is the maximum weight defined by weight function $w$. The above running time is upper bounded by $O(2.851^k 2^{o(k)} \log n \log W)$, by the similar analysis done for the $k$-Path problem in [12]. We output $\bigcup_{p \leq k} \widehat{\mathcal{H}_p}$ as the representative family corresponding to the node $v$. By Theorem 2.10, $|\widehat{\mathcal{H}_p}| \leq \binom{k}{p}2^{o(k)}$ and hence the number of subsets with $p$ elements in the representative family corresponding to $v$ is at most $\binom{k}{p}2^{o(k)}$. The computation corresponding to addition gate can be sped-up by using ideas given in [10].

**Multiplication Gate:** $v = v_1 \times v_2$**.** Similar to the previous case we have a representative families $\mathcal{F}_{v_1}$ and $\mathcal{F}_{v_2}$ for $v_1$ and $v_2$ respectively, where the number of subsets with $p$ elements in $\mathcal{F}_{v_1}$ as well as in $\mathcal{F}_{v_2}$, is at most $\binom{k}{p}2^{o(k)}$. Here, the representative family corresponding to $v$ will be the representative family of $\mathcal{F}_{v_1} \circ \mathcal{F}_{v_2}$. The idea is to get representative families using Corollary 3.4 for different values of $p_1$ and $p_2$. We have that

$$\mathcal{F}_{v_1} \circ \mathcal{F}_{v_2} = \bigcup_{p_1, p_2} \mathcal{F}_{v_1}^{p_1} \circ \mathcal{F}_{v_2}^{p_2},$$

where $\mathcal{F}_{v_i}^{p_i}$ contains all the subsets of size $p_i$ in $\mathcal{F}_{v_i}$. We know that $|\mathcal{F}_{v_i}^{p_i}| \leq \binom{k}{p_i}2^{o(k)}$. Now by using Corollary 3.4, we compute $\widehat{\mathcal{F}_{v_1}^{p_1} \circ \mathcal{F}_{v_2}^{p_2}} \subseteq_{minrep}^{k-p_1-p_2} \mathcal{F}_{v_1}^{p_1} \circ \mathcal{F}_{v_2}^{p_2}$ of size $\binom{k}{p_1+p_2} \cdot 2^{o(k)}$ for all $p_1, p_2$ such that $p_1 + p_2 \leq k$. Let $q = k - p_1 - p_2$, then all these computation can be done in time

$$\sum_{p_1,p_2} \min_{x_1,x_2} O\left( \frac{z(n,k,W)}{x_2^{p_1}(1-x_2)^{p_2}} + \frac{z(n,k,W)\cdot|\binom{k}{p_1}|}{x_1^{p_2}(1-x_1)^q(1-x_2)^{p_2}} + \frac{z(n,k,W)\cdot|\binom{k}{p_2}|}{x_1^{p_1}(1-x_1)^q x_2^{p_1}} \quad + \quad \frac{z(n,k,W)\cdot(\frac{k}{q})^q}{x_1^p(1-x_1)^q} \right).$$

Here, $z(n,k,W) = 2^{o(k)} n \log n \cdot \log W$. The above running time is upper bounded by $O(3.8408^k 2^{o(k)} \cdot n \log n \cdot \log W)$. We output $\bigcup_{p_1,p_2} \widehat{\mathcal{F}_{v_1}^{p_1} \circ \mathcal{F}_{v_2}^{p_2}}$ as the representative family corresponding to the node $v$. Note that the number of sets of size $p$ in $\bigcup_{p_1,p_2} \widehat{\mathcal{F}_{v_1}^{p_1} \circ \mathcal{F}_{v_2}^{p_2}}$ is bounded by $k \cdot \binom{k}{p} 2^{o(k)} \leq \binom{k}{p} 2^{o(k)}$.

Now we output a minimum weight set of size $k$ (if exists) among the representative family corresponding to the root node, otherwise we output No. Since there are $s(C)$ nodes in $C$, the total running time is bounded by $O(3.8408^k 2^{o(k)} s(C) n \log n \cdot \log W)$. This completes the proof. □

### 5.1 Matroidal Multilinear Monomial Detection

In this section we extend the $k$-wMLD problem to a matroidal version and design an algorithm for this. The problem Matroidal Multilinear Monomial Detection ($k$-wMMLD) is defined as follows.

| | |
|---|---|
| Matroidal Multilinear Monomial Detection | **Parameter:** $k$ |

**Input:** An arithmetic circuit $C$ over variables $X = \{x_1, x_2, \ldots, x_n\}$ representing a polynomial $P(X)$ over $\mathbb{Z}$, a linear matroid $M = (E, \mathcal{I})$ where the ground set $E = X$ with its representation matrix $A_M$ and an additive weight function $w : 2^X \to \mathbb{N}$.
**Question:** Does $P(X)$ construed as a sum of monomials contain a multilinear monomial $Z$ of degree $k$ such that $Z \in \mathcal{I}$? If yes find a minimum weighted such $Z$.

Our main theorem of this section is as follows. The proof of this theorem is along the lines of Theorem 5.2. The only difference is that we compute representative family with respect to the given matroid.

Theorem 5.3. *$k$-wMMLD can be solved in time $O(7.7703^k k^{\omega} s(C))$.*

Proof. Let $\pi = v_1, \ldots, v_n$ be a topological ordering of $C$ such that all the nodes corresponding to variables appear before any other gate and for every directed arc $uv$ we have that $u <_\pi v$. As in Theorem 5.2, at every node we keep a family $\mathcal{F}_{v_i}^j$ of *$j$-multilinear terms* that are also members of $\mathcal{I}$, where $j \in \{1, \ldots, k\}$. Let $\mathcal{F}_{v_i} = \bigcup_{x=1}^k \mathcal{F}_{v_i}^x$. So $\mathcal{F}_v \subseteq \mathcal{I}$. We process the nodes from left to right and keep $\widehat{\mathcal{F}_{v_i}^j} \subseteq_{minrep}^{k-j} \mathcal{F}_{v_i}^j$ of size $\binom{k}{p}$.

When $v$ is an input node then the associated family contains only one set. That is, if $v$ is labelled with $x_i$ and $\{x_i\} \in \mathcal{I}$ then $\mathcal{F}_v = \{\{x_i\}\}$, otherwise $\mathcal{F}_v = \{\emptyset\}$. If $v$ is labelled from $\mathbb{Z}^+$ then $\mathcal{F}_v = \{\emptyset\}$. When $v$ is not an input node, then we have two cases.

**Addition Gate:** $v = v_1 + v_2$. Due to the left to right computation in the topological order, we have representative families $\mathcal{F}_{v_1}$ and $\mathcal{F}_{v_2}$ for $v_1$ and $v_2$ respectively, where the number of subsets with $p$ elements in $\mathcal{F}_{v_1}$ as well as in $\mathcal{F}_{v_2}$ will be at most $\binom{k}{p}$. So the representative family corresponding to $v$ will be the representative family of $\mathcal{F}_{v_1} \cup \mathcal{F}_{v_2}$. We partition $\mathcal{F}_{v_1} \cup \mathcal{F}_{v_2}$ based on the size of subsets in it. Let $\mathcal{F}_{v_1} \cup \mathcal{F}_{v_2} = \biguplus_{p \leq k} \mathcal{H}_p$, where $\mathcal{H}_p$ contains all subsets of size $p$ in $\mathcal{F}_{v_1} \cup \mathcal{F}_{v_2}$. Note that $|\mathcal{H}_p| \leq 2\binom{k}{p}$. Now using Theorem 2.9 we can compute all $\widehat{\mathcal{H}_p} \subseteq_{minrep}^{k-p} \mathcal{H}_p$ in time

$$O\left(2 \sum_{p \leq k} \left\{ \binom{k}{p}\binom{k}{p} p^{\omega} + \binom{k}{p}\binom{k}{p}^{\omega-1} \right\}\right).$$

The above running time is upper bounded by $O(4^k p^{\omega} k + 2^{\omega k} k)$. We output $\bigcup_{p \le k} \widehat{\mathcal{H}_p}$ as the representative family corresponding to the node $v$. By Theorem 2.9, $|\widehat{\mathcal{H}_p}| \le \binom{k}{p}$ and thus the number of subsets with $p$ elements in $\bigcup_{p \le k} \widehat{\mathcal{H}_p}$ is at most $\binom{k}{p}$.

**Multiplication Gate:** $v = v_1 \times v_2$. Similar to the previous case we have a representative families $\mathcal{F}_{v_1}$ and $\mathcal{F}_{v_2}$ for $v_1$ and $v_2$ respectively, where the number of subsets with $p$ elements in $\mathcal{F}_{v_1}$ as well as in $\mathcal{F}_{v_2}$, is at most $\binom{k}{p}$. Here, the representative family corresponding to $v$ will be the representative family of $\mathcal{F}_{v_1} \bullet \mathcal{F}_{v_2}$. We have that

$$\mathcal{F}_{v_1} \bullet \mathcal{F}_{v_2} = \bigcup_{p_1, p_2} \mathcal{F}_{v_1}^{p_1} \bullet \mathcal{F}_{v_2}^{p_2},$$

where $\mathcal{F}_{v_i}^{p_i}$ contains all the subsets of size $p_i$ in $\mathcal{F}_{v_i}$. We know that $|\mathcal{F}_{v_i}^{p_i}| \le \binom{k}{p_i}$. Now by using Corollary 4.3, we can compute $\widehat{\mathcal{F}_{v_1}^{p_1} \bullet \mathcal{F}_{v_2}^{p_2}} \subseteq_{minrep}^{k-p_1-p_2} \mathcal{F}_{v_1}^{p_1} \bullet \mathcal{F}_{v_2}^{p_2}$ of size $\binom{k}{p_1+p_2}$ for all $p_1, p_2$ together in time $O\left(k^{\omega}\left(2^{\omega}+2\right)^k + k^{\omega} 2^{k(\omega-1)} 3^k\right)$.

Now let $\mathcal{F} = \bigcup_{p_1, p_2} \widehat{\mathcal{F}_{v_1}^{p_1} \bullet \mathcal{F}_{v_2}^{p_2}} = \uplus_p \mathcal{H}_p$, where $\uplus_p \mathcal{H}_p$ is the partition of $\mathcal{F}$ based on the size of subsets. It is easy to see that $|\mathcal{H}_p| \le k\binom{k}{p}$. Now using Theorem 2.9 we can compute $\widehat{\mathcal{H}_p} \subseteq_{minrep}^{k-p} \mathcal{H}_p$ for all $p \le k$ together in time

$$O\left(k \sum_{p \le k} \left\{ \binom{k}{p}\binom{k}{p} p^{\omega} + \binom{k}{p}\binom{k}{p}^{\omega-1} \right\}\right)$$

The above running time is upper bounded by $O(4^k k^{\omega+1} + 2^{\omega k} k^2)$. We output $\bigcup_{p \le k} \widehat{\mathcal{H}_p}$ as the representative family corresponding to the node $v$.

Now we output a minimum weight set of size $k$ (if exists) among the representative family corresponding to the root node, otherwise we output No. Since there are $s(C)$ nodes in $C$, the total running time is $O\left(k^{\omega}\left(2^{\omega}+2\right)^k s(C) + k^{\omega} 2^{k(\omega-1)} 3^k s(C)\right)$. This completes the proof. □

## 6 APPLICATION II: DYNAMIC PROGRAMMING OVER GRAPHS OF BOUNDED TREEWIDTH

In this section we discuss deterministic algorithms for "connectivity problems" such as Steiner Tree, Feedback Vertex Set parameterized by the treewidth of the input graph. The algorithms are based on Theorem 2.9 and Corollary 4.3. The idea of designing deterministic algorithms for connectivity problems parameterized by the treewidth of the input graph based on fast computation of representative families was outlined in [10]. Here, we show how we can speed the method described in [10] using the fast computation of representative families for product families coming from a graphic matroid. The method described in this section gives the fastest known deterministic algorithms for most the connectivity problems parameterized by the treewidth. We exemplify the methods on Steiner Tree and Feedback Vertex Set.

### 6.1 Treewidth

Let $G$ be a graph. A *tree decomposition* of a graph $G$ is a pair $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ such that

- $\bigcup_{t \in V(\mathbb{T})} X_t = V(G)$,
- for every edge $xy \in E(G)$ there is a $t \in V(\mathbb{T})$ such that $\{x, y\} \subseteq X_t$, and
- for every vertex $v \in V(G)$ the subgraph of $\mathbb{T}$ induced by the set $\{t \mid v \in X_t\}$ is connected.

The *width* of a tree decomposition is $\max_{t \in V(\mathbb{T})} |X_t| - 1$ and the *treewidth* of $G$ is the minimum width over all tree decompositions of $G$ and is denoted by $\mathbf{tw}(G)$.

A tree decomposition $(\mathbb{T}, \mathcal{X})$ is called a *nice tree decomposition* if $\mathbb{T}$ is a tree rooted at some node $r$ where $X_r = \emptyset$, each node of $\mathbb{T}$ has at most two children, and each node is of one of the following kinds:

(1) **Introduce node**: a node $t$ that has only one child $t'$ where $X_t \supset X_{t'}$ and $|X_t| = |X_{t'}| + 1$.
(2) **Forget node**: a node $t$ that has only one child $t'$ where $X_t \subset X_{t'}$ and $|X_t| = |X_{t'}| - 1$.
(3) **Join node**: a node $t$ with two children $t_1$ and $t_2$ such that $X_t = X_{t_1} = X_{t_2}$.
(4) **Base node**: a node $t$ that is a leaf of $\mathbb{T}$, is different than the root, and $X_t = \emptyset$.

Notice that, according to the above definition, the root $r$ of $\mathbb{T}$ is either a forget node or a join node. It is well known that any tree decomposition of $G$ can be transformed into a nice tree decomposition maintaining the same width in linear time [16]. We use $G_t$ to denote the graph induced by the vertex set $\bigcup_{t'} X_{t'}$, where $t'$ ranges over all descendants of $t$, including $t$. By $E(X_t)$ we denote the edges present in $G[X_t]$. We use $H_t$ to denote the graph on vertex set $V(G_t)$ and the edge set $E(G_t) \setminus E(X_t)$. For clarity of presentation we use the term nodes to refer to the vertices of the tree $\mathbb{T}$.

### 6.2 Steiner Tree Parameterized By Treewidth

The problem we study in this subsection is defined below.

| Steiner Tree |
|---|
| **Input**: An undirected graph $G$ with a set of terminals $T \subseteq V(G)$, and a non-negative weight function $w : E(G) \to \mathbb{N}$. |
| **Task**: Find a subtree in $G$ of minimum weight spanning all vertices of $T$. |

Let $G$ be an input graph of the Steiner Tree problem. Throughout this section, we say that $E' \subseteq E(G)$ is a *solution* if the subgraph induced on this edge set is connected and it contains all the terminal vertices. We call $E' \subseteq E(G)$ an *optimal solution* if $E'$ is a solution of the minimum weight. Let $\mathscr{S}$ be a family of edge subsets such that every edge subset corresponds to an optimal solution. That is,

$$\mathscr{S} = \{E' \subseteq E(G) \mid E' \text{ is an optimal solution}\}.$$

Observe that any edge set in $\mathscr{S}$ induces a forest. We start with a few definitions that will be useful in explaining the algorithm. Let $(\mathbb{T}, \mathcal{X})$ be a tree decomposition of $G$ of width $\mathbf{tw}$. Let $t$ be a node of $V(\mathbb{T})$. By $\mathcal{S}_t$ we denote the family of edge subsets of $E(H_t)$, $\{E' \subseteq E(H_t) \mid G[E'] \text{ is a forest}\}$, that satisfies one of the following properties.

- $E'$ is a solution tree (that is, the subgraph induced on this edge set is connected and it contains all the terminal vertices).
- Every vertex of $(T \cap V(G_t)) \setminus X_t$ is incident with some edge from $E'$, and every connected component of the graph induced by $E'$ contains a vertex from $X_t$.

We call $\mathcal{S}_t$ a *family of partial solutions* for $t$. We denote by $K^t$ a complete graph on the vertex set $X_t$. For an edge subset $E^* \subseteq E(G)$ and bag $X_t$ corresponding to a node $t$, we define the following.

(1) Set $\partial^t(E^*) = X_t \cap V(E^*)$, the set of endpoints of $E^*$ in $X_t$.
(2) Let $G^*$ be the subgraph of $G$ on the vertex set $V(G)$ and the edge set $E^*$. Let $C'_1, \ldots, C'_\ell$ be the connected components of $G^*$ such that for all $i \in [\ell]$, $C'_i \cap X_t \neq \emptyset$. Let $C_i = C'_i \cap X_t$. Observe that $C_1, \ldots, C_\ell$ is a partition of $\partial^t(E^*)$. By $F_t(E^*)$ we denote a forest $\{Q_1, \ldots, Q_\ell\}$ where each $Q_i$ is an arbitrary spanning tree of $K^t[C_i]$. For an example, since $K^t[C_i]$ is a

complete graph we could take $Q_i$ as a star. The purpose of $F_t(E^*)$ is to keep track for the vertices in $C_i$ whether they were in the same connected component of $G^*$.

(3) We define $w(F_t(E^*)) = w(E^*)$.

Let $\mathcal{A}$ and $\mathcal{B}$ be two families of edge subsets of $E(G)$, then we define

$$\mathcal{A} \diamond \mathcal{B} = \{E_1 \cup E_2 \mid E_1 \in \mathcal{A} \wedge E_2 \in \mathcal{B} \wedge E_1 \cap E_2 = \emptyset \wedge G[E_1 \cup E_2] \text{ is a forest}\}.$$

With every node $t$ of $\mathbb{T}$, we associate a subgraph of $G$. In our case it will be $H_t$. For every node $t$, we keep a family of partial solutions for the graph $H_t$. That is, for every optimal solution $L \in \mathscr{S}$ and its intersection $L_t = E(H_t) \cap L$ with the graph $H_t$, we have some partial solution in the family that is "as good as $L_t$". More precisely, we have some partial solution, say $\hat{L}_t$ in our family such that $\hat{L}_t \cup L_R$ is also an optimum solution for the whole graph, where $L_R = L \setminus L_t$. As we move from one node $t$ in the decomposition tree to the next node $t'$ the graph $H_t$ changes to $H_{t'}$, and so does the set of partial solutions. The algorithm updates its set of partial solutions accordingly. Here matroids come into play: in order to bound the size of the family of partial solutions that the algorithm stores at each node we employ Theorem 2.9 and Corollary 4.3 for graphic matroids. More details are given in the proof of the following theorem, which is one of the main results in this section.

Theorem 6.1. *Let $G$ be an $n$-vertex graph given together with its tree decomposition of width* $\mathbf{tw}$. *Then* Steiner Tree *on $G$ can be solved in time*
$O\left(\left(1 + 2^{\omega-1} \cdot 3\right)^{\mathbf{tw}} \mathbf{tw}^{O(1)} n\right)$.

Proof. For every node $t$ of $\mathbb{T}$ and subset $Z \subseteq X_t$, we store a family of edge subsets $\widehat{\mathcal{S}}_t[Z] \subseteq \mathcal{S}_t$ of $H_t$ satisfying the following correctness invariant.

**Correctness Invariant:** For every $L \in \mathscr{S}$ we have the following. Let $L_t = E(H_t) \cap L$, $L_R = L \setminus L_t$, and $Z = \partial^t(L)$. Then there exists $\hat{L}_t \in \widehat{\mathcal{S}}_t[Z]$ such that $w(\hat{L}_t) \le w(L_t)$, $\hat{L} = \hat{L}_t \cup L_R$ is a solution, and $\partial^t(\hat{L}) = Z$. Observe that since $w(\hat{L}_t) \le w(L_t)$ and $L \in \mathscr{S}$, we have that $\hat{L} \in \mathscr{S}$.

We process the nodes of the tree $\mathbb{T}$ from base nodes to the root node while doing the dynamic programming. Throughout the process we maintain the correctness invariant, which will prove the correctness of the algorithm. However, our main idea is to use representative families to obtain $\widehat{\mathcal{S}}_t[Z]$ of small size. That is, given the set $\widehat{\mathcal{S}}_t[Z]$ (as a product of two families $\mathcal{A}$ and $\mathcal{B}$, i.e $\widehat{\mathcal{S}}_t[Z] = \mathcal{A} \diamond \mathcal{B}$) that satisfies the correctness invariant, we use Corollary 4.3 to obtain a subset $\widehat{\mathcal{S}}'_t[Z]$ of $\widehat{\mathcal{S}}_t[Z]$ that also satisfies the correctness invariant and has size upper bounded by $2^{|Z|}$ in total. More precisely, the number of partial solutions with $i$ connected components in $\widehat{\mathcal{S}}'_t[Z]$ is upper bounded by $\binom{|Z|}{|Z|-i} = \binom{|Z|}{i}$. Thus, we maintain the following size invariant.

**Size Invariant:** After node $t$ of $\mathbb{T}$ is processed by the algorithm, for every $Z \subseteq X_t$ we have that $|\widehat{\mathcal{S}}_t[Z, i]| \le \binom{|Z|}{i}$, where $\widehat{\mathcal{S}}_t[Z, i]$ is the partial solutions with $i$ connected components in $\widehat{\mathcal{S}}_t[Z]$.

The main ingredient of the dynamic programming algorithm for Steiner Tree is the use of Theorem 2.9 and Corollary 4.3 to compute $\widehat{\mathcal{S}}_t[Z]$, maintaining the size invariant. The next lemma shows how to implement it.

Lemma 6.2 (Product Shrinking Lemma). *Let $t$ be a node of $\mathbb{T}$, and let $Z \subseteq X_t$ be a set of size $k$. Let $\mathcal{P}$ and $\mathcal{Q}$ be two families of edge sets of $H_t$. Furthermore, let $\widehat{\mathcal{S}}_t[Z] = \mathcal{P} \diamond \mathcal{Q}$ be the family of edge subsets of $H_t$ satisfying the correctness invariant. If the number of edge sets with $i$ connected* *components in $\mathcal{P}$ as well as in $\mathcal{Q}$ is bounded by $\binom{k+c}{i}$ where $c$ is some fixed constant, then in time $O\left(k^{\omega}\left(2^{\omega}+2\right)^{k} n+k^{\omega} 2^{k(\omega-1)} 3^{k} n\right)$ we can compute $\widehat{\mathcal{S}}_t'[Z] \subseteq \widehat{\mathcal{S}}_t[Z]$ satisfying correctness and size invariants.*

Proof. We start by associating a matroid with the node $t$ and the set $Z \subseteq X_t$ as follows. We consider a graphic matroid $M=(E,\mathcal{I})$ on $K^t[Z]$. Here, the element set $E$ of the matroid is the edge set $E(K^t[Z])$ and the family of independent sets $\mathcal{I}$ consists of forests of $K^t[Z]$. Let $\mathcal{P}=\{A_1,\ldots,A_\ell\}$ and $\mathcal{Q}=\{B_1,\ldots,B_{\ell'}\}$. Let $\mathcal{L}_1=\{F_t(A_1),\ldots,F_t(A_\ell)\}$ and $\mathcal{L}_2=\{F_t(B_1),\ldots,F_t(B_{\ell'})\}$ be the set of forests in $K^t[Z]$ corresponding to the edge subsets in $\mathcal{P}$ and $\mathcal{Q}$, respectively. For $r\in\{1,2\}$ and $i\in\{1,\ldots,k-1\}$, let $\mathcal{L}_{r,i}$ be the family of forests of $\mathcal{L}_r$ with $i$ edges. Now we apply Corollary 4.3 and find $\widehat{\mathcal{L}_{1,i}\bullet\mathcal{L}_{2,j}} \subseteq_{minrep}^{k-1-i-j} \mathcal{L}_{1,i}\bullet\mathcal{L}_{2,j}$ of size $\binom{k-1}{i+j}$ for all $i,j\in[k]$ such that $i+j<k$. Let $\widehat{\mathcal{S}}_t'[Z,k-d]\subseteq\widehat{\mathcal{S}}_t[Z,k-d]$ be such that for every $D\in\widehat{\mathcal{S}}_t'[Z,k-d]$ we have that $F_t(D)\in\bigcup_{i+j=d}\widehat{\mathcal{L}_{1,i}\bullet\mathcal{L}_{2,j}}$. Note that $F_t(D)$ has $d$ edges if and only if $G[D]$ have $k-d$ connected components. Let $\widehat{\mathcal{S}}_t'[Z]=\bigcup_{j=1}^{k}\widehat{\mathcal{S}}_t'[Z,j]$. By Corollary 4.3, $|\widehat{\mathcal{S}}_t'[Z,k-d]|\le k\binom{k-1}{d}\le\binom{k}{k-d}$, and hence $\widehat{\mathcal{S}}_t'[Z]$ maintains the size invariant.

Now we show that the $\widehat{\mathcal{S}}_t'[Z]$ maintains the correctness invariant. Let $L\in\mathscr{S}$. Let $L_t=E(H_t)\cap L$, $L_R=L\setminus L_t$ and $Z=\partial^t(L)$. Since $\widehat{\mathcal{S}}_t[Z]$ satisfies correctness invariant, there exists $L_t'\in\widehat{\mathcal{S}}_t[Z]$ such that $w(L_t')\le w(L_t)$, $\hat{L}=L_t'\cup L_R$ is an optimal solution and $\partial^t(\hat{L})=Z$. Since $\widehat{\mathcal{S}}_t[Z]=\mathcal{P}\diamond\mathcal{Q}$, there exist $A\in\mathcal{P}$ and $B\in\mathcal{Q}$ such that $L_t'=A\cup B$. Observe that $G[L_t'],G[A]$ and $G[B]$ form forests. Consider the forests $F_t(A)$ and $F_t(B)$. Suppose $F_t(A)$ has $i$ edges and $F_t(B)$ has $j$ edges, then $F_t(L_t')\in\mathcal{L}_{1,i}\bullet\mathcal{L}_{2,j}$. This is because if $F_t(L_t')$ contain a cycle, then corresponding to that cycle we can get a cycle in $G[L_t']$, which is a contradiction. Now let $F_t(L_R)$ be the forest corresponding to $L_R$. Since $\hat{L}$ is a solution, we have that $F_t(L_t')\cup F_t(L_R)$ is a spanning tree in $K^t[Z]$. Since $\widehat{\mathcal{L}_{1,j}\bullet\mathcal{L}_{2,j}}\subseteq_{minrep}^{k-1-i-j}\mathcal{L}_{1,i}\bullet\mathcal{L}_{2,j}$, we have that there exists a forest $F_t(\widehat{L}_t')\in\widehat{\mathcal{L}_{1,i}\bullet\mathcal{L}_{2,j}}$ such that $w(F_t(\widehat{L}_t'))\le w(F_t(L_t'))$ and $F(\widehat{L}_t')\cup F(L_R)$ is a spanning tree in $K^t[Z]$. Thus, we have that $\widehat{L}_t'\cup L_R$ is an optimum solution and $\widehat{L}_t'\in\widehat{\mathcal{S}}_t'[Z]$. This proves that $\widehat{\mathcal{S}}_t'[Z]$ maintains the correctness invariant.

For a given edge set $D$, we need to compute the forest $F_t(D)$ and that can take $O(n)$ time. The running time to compute $\widehat{\mathcal{S}}_t'[Z]$ is,

$$O\left(k^{\omega}\left(2^{\omega}+2\right)^{k} n+k^{\omega} 2^{k(\omega-1)} 3^{k} n\right).$$

This completes the proof of the lemma. □

We now return to the dynamic programming algorithm over the tree decomposition $(\mathbb{T},\mathcal{X})$ of $G$ and prove that it maintains the correctness invariant. We assume that $(\mathbb{T},\mathcal{X})$ is a nice tree decomposition of $G$. By $\widehat{\mathcal{S}}_t$ we denote $\bigcup_{Z\subseteq X_t}\widehat{\mathcal{S}}_t[Z]$ (also called a *representative family of partial solutions*). We show how $\widehat{\mathcal{S}}_t$ is obtained by doing dynamic programming from base node to the root node.

*Base node $t$.* Here the graph $H_t$ is empty and thus we take $\widehat{\mathcal{S}}_t=\{\emptyset\}$.

*Introduce node $t$ with child $t'$.* Here, we know that $X_t\supset X_{t'}$ and $|X_t|=|X_{t'}|+1$. Let $v$ be the vertex in $X_t\setminus X_{t'}$. Furthermore, observe that $E(H_t)=E(H_{t'})$ and $v$ is a degree zero vertex in $H_t$. Thus the graph $H_t$ only differs from $H_{t'}$ at an isolated vertex $v$. Since we have not added any edge to the new graph, the family of solutions, which contains edge-subsets, does not change. Thus, we take $\widehat{\mathcal{S}}_t=\widehat{\mathcal{S}}_{t'}$. Formally, we take $\widehat{\mathcal{S}}_t[Z]=\widehat{\mathcal{S}}_{t'}[Z\setminus\{v\}]$. Since $H_t$ and $H_{t'}$ have same set of edges the invariant is vacuously maintained.

*Forget node $t$ with child $t'$.* Here we know $X_t \subset X_{t'}$ and $|X_t| = |X_{t'}| - 1$. Let $v$ be the vertex in $X_{t'} \setminus X_t$. Let $\mathcal{E}_v[Z]$ denote the set of edges between $v$ and the vertices in $Z \subseteq X_t$. Observe that $E(H_t) = E(H_{t'}) \cup \mathcal{E}_v[X_t]$. Before we define things formally, observe that in this step the graphs $H_t$ and $H_{t'}$ differ by at most **tw** edges - the edges with one endpoint in $v$ and the other in $X_t$. We go through every possible way an optimal solution can intersect with these newly added edges. Let $\mathcal{P}_v[Z] = \{Y \mid \emptyset \neq Y \subseteq \mathcal{E}_v[Z]\}$. Then the new set of partial solutions is defined as follows.

$$\widehat{\mathcal{S}}_t[Z] = \begin{cases} \left(\widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \diamond \mathcal{P}_v[Z]\right) \cup \left\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\right\} & \text{if } v \in T \\ \left(\widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \diamond \mathcal{P}_v[Z]\right) \cup \left\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\right\} \cup \widehat{\mathcal{S}}_{t'}[Z] & \text{if } v \notin T \end{cases}$$

Now we claim that $\widehat{\mathcal{S}}_t[Z] \subseteq \mathcal{S}_t$. Towards the proof we first show that $\widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \diamond \mathcal{P}_v[Z] \subseteq \mathcal{S}_t$. Let $E' \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \diamond \mathcal{P}_v[Z]$. Note that $E' \cap \mathcal{E}_v[Z] \neq \emptyset$. If $E'$ is a solution tree then $E' \in \mathcal{S}_t$ and we are done. Since $E' \setminus \mathcal{E}_v[Z] \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \subseteq \mathcal{S}_{t'}$, every vertex of $(T \cap V(G_t)) \setminus (X_t \cup \{v\})$ is incident with some edge from $E'$. Since $E' \cap \mathcal{E}_v[Z] \neq \emptyset$, there exists an edge in $E'$ which is incident to $v$. This implies that every vertex of $(T \cap V(G_t)) \setminus X_t$ is incident with some edge from $E'$. Now consider any connected component $C$ in $G[E']$. If $v \notin V(C)$, then $C$ contains a vertex from $X_{t'} \setminus \{v\} = X_t$, because $E' \setminus \mathcal{E}_v[Z] \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \subseteq \mathcal{S}_{t'}$. If $v \in V(C)$, then $C$ contains a vertex from $X_t$ because $E' \cap \mathcal{E}_v[Z] \neq \emptyset$. Thus we have shown that $E' \in \mathcal{S}_t$. It is easy to see that $\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\} \subseteq \mathcal{S}_t$. If $v \notin T$ then $\widehat{\mathcal{S}}_{t'}[Z] \subseteq \mathcal{S}_t$, because $\widehat{\mathcal{S}}_{t'}[Z] \subseteq \mathcal{S}_{t'}$ and $X_t = X_{t'} \setminus \{v\}$.

Now we show that $\widehat{\mathcal{S}}_t$ maintains the invariant of the algorithm. Let $L \in \mathscr{S}$.

(1) Let $L_t = E(H_t) \cap L$ and $L_R = L \setminus L_t$. Furthermore, edges of $L_t$ can be partitioned into $L_{t'} = E(H_{t'}) \cap L$ and $L_v = L_t \setminus L_{t'}$. That is, $L_t = L_{t'} \uplus L_v$.
(2) Let $Z = \partial^t(L)$ and $Z' = \partial^{t'}(L)$.

By the property of $\widehat{\mathcal{S}}_{t'}$, there exists a $\hat{L}_{t'} \in \widehat{\mathcal{S}}_{t'}[Z']$ such that

$$\begin{aligned} L \in \mathscr{S} \quad &\Longleftrightarrow \quad L_{t'} \uplus L_v \uplus L_R \in \mathscr{S} \\ &\Longleftrightarrow \quad \hat{L}_{t'} \uplus L_v \uplus L_R \in \mathscr{S} \end{aligned} \tag{1}$$

and $\partial^{t'}(L) = \partial^{t'}(\hat{L}_{t'} \uplus L_v \uplus L_R) = Z'$.

We put $\hat{L}_t = \hat{L}_{t'} \cup L_v$ and $\hat{L} = \hat{L}_t \cup L_R$. We now show that $\hat{L}_t \in \widehat{\mathcal{S}}_t[Z]$. If $v \notin Z'$, then $v \notin T$, $\hat{L}_t = \hat{L}_{t'}$ and $Z = Z'$. This implies that $\hat{L}_t \in \widehat{\mathcal{S}}_t[Z]$. If $v \in Z'$ and $L_v \neq \emptyset$ then $Z' = Z \cup \{v\}$. This implies that $\hat{L}_t \in \widehat{\mathcal{S}}_{t'}[Z'] \diamond \{L_v\} \subseteq \widehat{\mathcal{S}}_t[Z]$. If $v \in Z'$ and $L_v = \emptyset$ then $Z' = Z \cup \{v\}$ and $\hat{L}_t = \hat{L}_{t'}$. This implies that $\hat{L}_t \in \{A \in \widehat{\mathcal{S}}_{t'}[Z'] : A \in \mathcal{S}_t\} \subseteq \widehat{\mathcal{S}}_t[Z]$. By (1), $\hat{L} \in \mathscr{S}$. Finally, we need to show that $\partial^t(\hat{L}) = Z$. Towards this just note that $\partial^t(\hat{L}) = Z' \setminus \{v\} = Z$. This concludes the proof for the fact that $\widehat{\mathcal{S}}_t$ maintains the correctness invariant.

*Join node $t$ with two children $t_1$ and $t_2$.* Here, we know that $X_t = X_{t_1} = X_{t_2}$. Also we know that the edges of $H_t$ is obtained by the union of edges of $H_{t_1}$ and $H_{t_2}$ which are disjoint. Of course they are separated by the vertices in $X_t$. A natural way to obtain a family of partial solutions for $H_t$ is that we take the union of edge subsets of the families stored at nodes $t_1$ and $t_2$. This is exactly what we do. Let

$$\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t_1}[Z] \diamond \widehat{\mathcal{S}}_{t_2}[Z].$$

Now we show that $\widehat{\mathcal{S}}_t$ maintains the invariant. Let $L \in \mathscr{S}$.

(1) Let $L_t = E(H_t) \cap L$ and $L_R = L \setminus L_t$. Furthermore, edges of $L_t$ can be partitioned into those belonging to $H_{t_1}$ and those belonging to $H_{t_2}$. Let $L_{t_1} = E(H_{t_1}) \cap L$ and $L_{t_2} = E(H_{t_2}) \cap L$.

Observe that since $E(H_{t_1}) \cap E(H_{t_2}) = \emptyset$, we have that $L_{t_1} \cap L_{t_2} = \emptyset$. Also observe that $L_t = L_{t_1} \uplus L_{t_2}$ and $G[L_{t_1}], G[L_{t_1}]$ form forests.

(2) Let $Z = \partial^t(L)$. Since $X_t = X_{t_1} = X_{t_2}$ this implies that $Z = \partial^t(L) = \partial^{t_1}(L) = \partial^{t_2}(L)$.

Now observe that

$$\begin{aligned}
L \in \mathscr{S} \quad &\iff \quad L_{t_1} \uplus L_{t_2} \uplus L_R \in \mathscr{S} \\
&\iff \quad \hat{L}_{t_1} \uplus L_{t_2} \uplus L_R \in \mathscr{S} \text{ (by the property of } \widehat{\mathcal{S}}_{t_1} \text{ we have that } \hat{L}_{t_1} \in \widehat{\mathcal{S}}_{t_1}[Z]) \\
&\iff \quad \hat{L}_{t_1} \uplus \hat{L}_{t_2} \uplus L_R \in \mathscr{S} \text{ (by the property of } \widehat{\mathcal{S}}_{t_2} \text{ we have that } \hat{L}_{t_2} \in \widehat{\mathcal{S}}_{t_2}[Z])
\end{aligned}$$

We put $\hat{L}_t = \hat{L}_{t_1} \cup \hat{L}_{t_2}$. By the definition of $\widehat{\mathcal{S}}_t[Z]$, we have that $\hat{L}_{t_1} \cup \hat{L}_{t_2} \in \widehat{\mathcal{S}}_t[Z]$. The above inequalities also show that $\hat{L} = \hat{L}_t \cup L_R \in \mathscr{S}$. It remains to show that $\partial^t(\hat{L}) = Z$. Since $\partial^{t_1}(L) = Z$, we have that $\partial^{t_1}(\hat{L}_{t_1} \uplus L_{t_2} \uplus L_R) = Z$. Now since $X_{t_1} = X_{t_2}$ we have that $\partial^{t_2}(\hat{L}_{t_1} \uplus L_{t_2} \uplus L_R) = Z$ and thus $\partial^{t_2}(\hat{L}_{t_1} \uplus \hat{L}_{t_2} \uplus L_R) = Z$. Finally, because $X_{t_2} = X_t$, we conclude that $\partial^t(\hat{L}_{t_1} \uplus \hat{L}_{t_2} \uplus L_R) = \partial^t(\hat{L}) = Z$. This concludes the proof of correctness invariant.

*Root node $r$.* Here, $X_r = \emptyset$. We go through all the solution in $\widehat{\mathcal{S}}_r[\emptyset]$ and output the one with the minimum weight. This concludes the description of the dynamic programming algorithm.

**Computation of $\widehat{\mathcal{S}}_t$.** Now we show how to implement the algorithm described above in the desired running time by making use of Lemma 6.2. For our discussion let us fix a node $t$ and $Z \subseteq X_t$ of size $k$. While doing dynamic programming algorithm from the base nodes to the root node we always maintain the size invariant.

*Base node $t$.* Trivially, in this case we have maintained the size invariant.

*Introduce node $t$ with child $t'$.* Here, we have that $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t'}[Z \setminus \{v\}]$ and thus the number of partial solutions with $i$ connected components in $\widehat{\mathcal{S}}_t[Z]$ is bounded $\binom{k}{i}$.

*Forget node $t$ with child $t'$.* In this case,

$$\widehat{\mathcal{S}}_t[Z] = \begin{cases} \left(\widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \diamond \mathcal{P}_v[Z]\right) \cup \left\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\right\} & \text{if } v \in T \\ \left(\widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \diamond \mathcal{P}_v[Z]\right) \cup \left\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\right\} \cup \widehat{\mathcal{S}}_{t'}[Z] & \text{if } v \notin T \end{cases}$$

Since $\widehat{\mathcal{S}}_{t'}[Z \cup \{v\}]$ maintains size invariant, the number of edge subsets with $i$ connected components in $\widehat{\mathcal{S}}_{t'}[Z \cup \{v\}]$ is upper bounded by $\binom{k+1}{i}$. It is easy to see that the number of edge subsets with $i$ connected components in $\mathcal{P}_v[Z]$ is upper bounded by $\binom{k}{i}$. So first we apply Lemma 6.2 and obtain $\mathcal{R} \subseteq \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \diamond \mathcal{P}_v[Z]$ that maintains the correctness and size invariants. Now let

$$\widehat{\mathcal{S}}'_t[Z] = \begin{cases} \mathcal{R} \cup \left\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\right\} & \text{if } v \in T \\ \mathcal{R} \cup \left\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\right\} \cup \widehat{\mathcal{S}}_{t'}[Z] & \text{if } v \notin T \end{cases}$$

Note that $\widehat{\mathcal{S}}'_t[Z]$ maintains correctness invariant. Since the number of edge subsets with $i$ connected components in $\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\}$ and $\widehat{\mathcal{S}}_{t'}[Z]$ is bounded by $\binom{k+1}{i}$, the the number of edge subsets with $i$ connected components in $\widehat{\mathcal{S}}'_t[Z]$ is at most $\binom{k+4}{i}$. Also note that $\widehat{\mathcal{S}}'_t[Z] = \widehat{\mathcal{S}}'_t[Z] \diamond \{\emptyset\}$. Thus we can apply Lemma 6.2 and obtain $\widehat{\mathcal{S}}''_t[Z] \subseteq \widehat{\mathcal{S}}'_t[Z]$ that maintains the correctness and size invariants. We update $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}''_t[Z]$.

The running time to compute $\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\}$ is $O(2^{|Z|}n)$. Thus the running time $T$ to compute $\widehat{\mathcal{S}}_t$ (that is, across all subsets of $X_t$) is

$$\begin{aligned} T &= O\left(\sum_{i=1}^{\mathbf{tw}+1}\binom{\mathbf{tw}+1}{i}\left(i^{\omega}\left(2^{\omega}+2\right)^{i}n + i^{\omega}2^{i(\omega-1)}3^{i}n\right) + \sum_{i=1}^{\mathbf{tw}+1}\binom{\mathbf{tw}+1}{i}2^{i}n\right) \\ &= O\left(\mathbf{tw}^{\omega}n\left(2^{\omega}+3\right)^{\mathbf{tw}} + \mathbf{tw}^{\omega}n\left(1+2^{\omega-1}\cdot 3\right)^{\mathbf{tw}}\right) \end{aligned}$$

*Join node $t$ with two children $t_1$ and $t_2$.* Here we defined

$$\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t_1}[Z] \diamond \widehat{\mathcal{S}}_{t_2}[Z].$$

The number of edge subsets with $i$ connected components in $\widehat{\mathcal{S}}_{t_1}[Z]$ and $\widehat{\mathcal{S}}_{t_2}[Z]$ are bounded by $\binom{k}{i}$. Now, we apply Lemma 6.2 and obtain $\widehat{\mathcal{S}}'_t[Z]$ that maintains the correctness invariant and has size at most $2^k$. We put $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}'_t[Z]$. The running time to compute $\widehat{\mathcal{S}}_t$ is

$$O\left(\mathbf{tw}^{\omega}n\left(2^{\omega}+3\right)^{\mathbf{tw}} + \mathbf{tw}^{\omega}n\left(1+2^{\omega-1}\cdot 3\right)^{\mathbf{tw}}\right).$$

Thus the whole algorithm takes $O\left(\mathbf{tw}^{\omega}n^2\left(2^{\omega}+3\right)^{\mathbf{tw}} + \mathbf{tw}^{\omega}n^2\left(1+2^{\omega-1}\cdot 3\right)^{\mathbf{tw}}\right) = O(8.7703^{\mathbf{tw}}n^2)$ as the number of nodes in a nice tree-decomposition is upper bounded by $O(n)$. However, observe that we do not need to compute the forests and the associated weight at every step of the algorithm. The size of the forest is at most $\mathbf{tw}+1$ and we can maintain these forests across the bags during dynamic programming in time $\mathbf{tw}^{O(1)}$. Also, these forests can be used to compute the set $\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\}$ during the computation in the forget node $t$. This will lead to an algorithm with the claimed running time. This completes the proof. ◻

### 6.3 Feedback Vertex Set Parameterized By Treewidth

In this subsection we study the Feedback Vertex Set problem which is defined as follows.

| |
|---|
| Feedback Vertex Set<br>**Input**: An undirected graph $G$ and a non negative weight function $w : V(G) \to \mathbb{N}$.<br>**Task**: Find a minimum weight set $Y \subseteq V(G)$ such that $G[V(G) \setminus Y]$ is a forest. |

Let $G$ be an input graph of the Feedback Vertex Set problem. In this subsection instead of saying feedback vertex set $Y \subseteq V(G)$ is a solution, we say that $V(G) \setminus Y$ is a solution, i.e, our objective is to find a maximum weight set $V' \subseteq V(G)$ such that $G[V']$ is a forest. We call $V' \subseteq V(G)$ is an *optimal solution* if $V'$ is a solution with maximum weight. Let $\mathscr{S}$ be a family of vertex subsets such that every vertex subset corresponds to an optimal solution. That is,

$$\mathscr{S} = \{V' \subseteq V(G) \mid V' \text{ is an optimal solution}\}.$$

Let $(\mathbb{T}, \mathcal{X})$ be a tree decomposition of $G$ of width $\mathbf{tw}$. For each tree node $t$ and $Z \subseteq X_t$, we define $\mathcal{S}_t[Z]$, *family of partial solutions* as follows.

$$\mathcal{S}_t[Z] = \{U \subseteq V(H_t) \mid U \cap X_t = Z \text{ and } H_t[U] \text{ is a forest }\}$$

We denote by $K^t$ a complete graph on the vertex set $X_t$. Let $G^*$ be subgraph of $G$. Let $C'_1, \ldots, C'_\ell$ be the connected components of $G^*$ that have nonempty intersection with $X_t$. Let $C_i = C'_i \cap X_t$. By $F_t(G^*)$ we denote the a forest $\{Q_1, \ldots, Q_\ell\}$ where each $Q_i$ is an arbitrary spanning tree of $K^t[C_i]$.

For two family of vertex subsets $\mathcal{P}$ and $\mathcal{Q}$ of the subgraph $H_t$, we denote

$$\mathcal{P} \otimes_t \mathcal{Q} = \{U_1 \cup U_2 \mid U_1 \in \mathcal{P}, U_2 \in \mathcal{Q} \text{ and } H_t[U_1 \cup U_2] \text{ is a forest } \}.$$

With every node $t$ of $\mathbb{T}$, we associate the subgraph $H_t$ of $G$. For every node $t$, we keep a family of partial solutions for the graph $H_t$ which is sufficient to guarantee the correctness of the algorithm. That is for every optimal solution $L \in \mathscr{S}$ with $L \cap X_t = Z$ and its intersection $L_t = V(H_t) \cap L$ with the graph $H_t$, we have some partial solution $\hat{L_t}$ in our subset such that $\hat{L_t} \cap X_t = Z$ and $\hat{L_t} \cup L_R$ is an optimal solution, i.e $G[\hat{L_t} \cup L_R]$ is a forest, where $L_R = L \setminus L_t$ and $w(\hat{L_t} \cup L_R) \geq w(L)$. Now we are ready to state the main theorem.

Theorem 6.3. *Let $G$ be an $n$-vertex graph given together with its tree decomposition of width* **tw**. *Then* Feedback Vertex Set *on $G$ can be solved in time $O\left(\left(1 + 2^{\omega-1} \cdot 3\right)^{\mathbf{tw}} \mathbf{tw}^{O(1)} n\right)$.*

Proof. For every node $t$ of $\mathbb{T}$ and $Z \subseteq X_t$, we store a family of vertex subsets $\widehat{\mathcal{S}}_t[Z]$ of $V(H_t)$ satisfying the following correctness invariant.

> **Correctness Invariant:** For every $L \in \mathscr{S}$ we have the following. Let $L_t = V(H_t) \cap L$, $L_R = L \setminus L_t$ and $L \cap X_t = Z$. Then there exists $\hat{L}_t \in \widehat{\mathcal{S}}_t[Z]$ such that $\hat{L} = \hat{L}_t \cup L_R$ is an optimal solution, i.e $G[\hat{L}_t \cup L_R]$ is a forest with $w(\hat{L}_t) \geq w(L_t)$. Thus we have that $\hat{L} \in \mathscr{S}$.

We process the nodes of the tree $\mathbb{T}$ from base nodes to the root node while doing the dynamic programming. Throughout the process we maintain the correctness invariant, which will prove the correctness of the algorithm. However, our main idea is to use representative families to obtain $\widehat{\mathcal{S}}_t[Z]$ of small size. That is, given the set $\widehat{\mathcal{S}}_t[Z]$ that satisfies the correctness invariant, we use *representative family* tool to obtain a subset $\widehat{\mathcal{S}}'_t[Z]$ of $\widehat{\mathcal{S}}_t[Z]$ that also satisfies the correctness invariant and has size upper bounded by $2^{|Z|}$ in total. More precisely, the number of partial solutions in $\widehat{\mathcal{S}}'_t[Z]$ that have $i$ connected components with nonempty intersection with $X_t$ is upper bounded by $\binom{|Z|}{i}$. Thus, we maintain the following size invariant.

> **Size Invariant:** After node $t$ of $\mathbb{T}$ is processed by the algorithm, we have that $|\widehat{\mathcal{S}}_t[Z, i]| \leq \binom{|Z|}{i}$, where $\widehat{\mathcal{S}}_t[Z, i]$ is the set of partial solutions that have $i$ connected components with nonempty intersection with $X_t$.

Lemma 6.4 (Product Shrinking Lemma). *Let $t$ be a node of $\mathbb{T}$ and let $Z \subseteq X_t$ be a set of size $k$. Let $\mathcal{P}$ and $\mathcal{Q}$ be two families of vertex subsets of $V(H_t)$ (partial solutions) such that for any $A \in \mathcal{P}$ and $B \in \mathcal{Q}$, $E(H_t[A]) \cap E(H_t[B]) = \emptyset$. Furthermore, let $\widehat{\mathcal{S}}_t[Z] = \mathcal{P} \otimes_t \mathcal{Q}$ be the family of vertex subsets of $V(H_t)$ satisfying the correctness invariant. If the number of partial solutions with $i$ connected components having nonempty intersection with $Z$ in $\mathcal{P}$ as well as in $\mathcal{Q}$ is bounded by $\binom{k+c}{i}$ where $c$ is some fixed constant, then in time $O\left(k^{\omega}\left(2^{\omega}+2\right)^{k} n + k^{\omega} 2^{k(\omega-1)} 3^{k} n\right)$ we can compute $\widehat{\mathcal{S}}'_t[Z] \subseteq \widehat{\mathcal{S}}_t[Z]$ satisfying correctness and size invariants.*

Proof. We start by associating a matroid with node $t$ and the set $Z \subseteq X_t$ as follows. We consider a graphic matroid $M = (E, \mathcal{I})$ on $K^t[Z]$. Here, the element set $E$ of the matroid is the edge set $E(K^t[Z])$ and the family of independent sets $\mathcal{I}$ consists of spanning forests of $K^t[Z]$. Here our objective is to find a small subfamily of $\widehat{\mathcal{S}}_t[Z] = \mathcal{P} \otimes_t \mathcal{Q}$ satisfying correctness and size invariants using efficient computation of representative family in the graphic matroid $M$. The main idea to prune the size of partial solutions is as follows: for each independent set $U \in \widehat{\mathcal{S}}_t[Z]$ we associate $F_t(H_t[U])$ as the corresponding independent set in the graphic matroid $M$ and compute representative family in the graphic matroid $M$.

Let $\mathcal{P} = \{A_1, \ldots, A_\ell\}$ and $\mathcal{Q} = \{B_1, \ldots, B_{\ell'}\}$. Let $\mathcal{L}_1 = \{F_t(H_t[A_1]), \ldots, F_t(H_t[A_\ell])\}$ and $\mathcal{L}_2 = \{F_t(H_t[B_1]), \ldots, F_t(H_t[B_{\ell'}])\}$ be the set of forests in $K^t[Z]$ corresponding to the vertex subsets in $\mathcal{P}$ and $\mathcal{Q}$ respectively. Now we define a non negative weight function $w' : \mathcal{L}_1 \bullet \mathcal{L}_2 \rightarrow \mathbb{N}$ as follows. For each $F_t(H_t[A_i]) \cup F_t(H_t[B_j]) \in \mathcal{L}_1 \bullet \mathcal{L}_2$ we set $w'\left(F_t(H_t[A_i]) \cup F_t(H_t[B_j])\right) = w(A_i \cup B_j)$. For $i \in [k]$ and $r \in \{1, 2\}$, let $\mathcal{L}_{r,i}$ be the family of forests of $\mathcal{L}_r$ with $i$ edges. Now we apply Corollary 4.3 and find $\widehat{\mathcal{L}_{1,i} \bullet \mathcal{L}_{2,j}} \subseteq^{k-1-i-j}_{maxrep} \mathcal{L}_{1,i} \bullet \mathcal{L}_{2,j}$ of size $\binom{k-1}{i+j}$ for all $i, j \in [k]$. Let $\widehat{\mathcal{S}}'_t[Z, k-d] \subseteq \widehat{\mathcal{S}}_t[Z, k-d]$ be such that for every $U_1 \cup U_2 \in \widehat{\mathcal{S}}'_t[Z, k-d]$ we have that $F_t(H_t[U_1]) \cup F_t(H_t[U_2]) \in \bigcup_{i+j=d} \widehat{\mathcal{L}_{1,i} \bullet \mathcal{L}_{2,j}}$. Let $\widehat{\mathcal{S}}'_t[Z] = \bigcup_{j=0}^{k} \widehat{\mathcal{S}}'_t[Z, j]$. By Corollary 4.3, $|\widehat{\mathcal{S}}'_t[Z, k-d]| \leq k\binom{k-1}{d} \leq \binom{k}{k-d}$, and hence $\widehat{\mathcal{S}}'_t[Z]$ maintains the size invariant.

Now we show that the $\widehat{\mathcal{S}}'_t[Z]$ maintains the correctness invariant. Let $L \in \mathscr{S}$ and let $L_t = V(H_t) \cap L$, $L_R = L \setminus L_t$ and $Z = L \cap X_t$. Since $\widehat{\mathcal{S}}_t[Z]$ satisfy correctness invariant, there exists $\hat{L}_t \in \widehat{\mathcal{S}}_t[Z]$ such that $w(\hat{L}_t) \geq w(L_t)$, $\hat{L} = \hat{L}_t \cup L_R$ is an optimal solution and $\hat{L} \cap X_t = Z$. Since $\widehat{\mathcal{S}}_t[Z] = \mathcal{P} \otimes_t \mathcal{Q}$, there exists $U_1 \in \mathcal{P}$ and $U_2 \in \mathcal{Q}$ such that $\hat{L}_t = U_1 \cup U_2$. Observe that $H_t[U_1 \cup U_2]$ form a forest. Consider the forests $F_t(H_t[U_1])$ and $F_t(H_t[U_2])$. Suppose $|F_t(H_t[U_1])| = i_1$ and $|F_t(H_t[U_2])| = i_2$, then $F_t(H_t[U_1]) \cup F_t(H_t[U_2]) \in \mathcal{L}_{1,i_1} \bullet \mathcal{L}_{1,i_2}$. This is because if $F_t(H_t[U_1]) \cup F_t(H_t[U_2])$ contains a cycle, then corresponding to that cycle we can get a cycle in $H_t[U_1 \cup U_2]$, which is a contradiction. Now let $E' = F_t(G[L_R \cup Z])$ be the forest corresponding to $L_R \cup Z$ with respect to the bag $X_t$. Since $\hat{L}$ is a solution, we have that $F_t(H_t[U_1]) \cup F_t(H_t[U_2]) \cup E'$ is a forest in $K^t[Z]$. Since $\widehat{\mathcal{L}_{1,i_1} \bullet \mathcal{L}_{2,i_2}} \subseteq^{k-1-i_1-i_2}_{maxrep} \mathcal{L}_{1,i_1} \bullet \mathcal{L}_{2,i_2}$, there exists a forest $F_t(H_t[U'_1]) \cup F_t(H_t[U'_2]) \in \widehat{\mathcal{L}_{1,i_1} \bullet \mathcal{L}_{2,i_2}}$ such that $w'\left(F_t(H_t[U'_1]) \cup F_t(H_t[U'_2])\right) \geq w'\left(F_t(H_t[U_1] \cup F_t(H_t[U_2]))\right) = w(U_1 \cup U_2)$ and $F_t(H_t[U'_1]) \cup F_t(H_t[U'_2]) \cup E'$ is a forest in $K^t[Z]$. Hence $U'_1 \cup U'_2 \in \widehat{\mathcal{S}}'_t[Z]$. Since $w(U'_1 \cup U'_2) = w'\left(F_t(H_t[U'_1]) \cup F_t(H_t[U'_2])\right)$, $w(U'_1 \cup U'_2) \geq w(U_1 \cup U_2)$. Thus, we can conclude that $U'_1 \cup U'_2 \cup L_R$ is an optimal solution. This proves that $\widehat{\mathcal{S}}'_t[Z]$ maintains the correctness invariant.

By Corollary 4.3, the running time to compute $\widehat{\mathcal{S}}'_t[Z]$ is upper bounded by,

$$\mathcal{O}\left(k^{\omega}\left(2^{\omega}+2\right)^{k} n + k^{\omega} 2^{k(\omega-1)} 3^{k} n\right).$$

This completes the proof of the lemma. □

We now explain the dynamic programming algorithm over the tree-decomposition $(\mathbb{T}, \mathcal{X})$ of $G$ and prove that it maintains the correctness invariant. We assume that $(\mathbb{T}, \mathcal{X})$ is a nice tree-decomposition of $G$. By $\widehat{\mathcal{S}}_t$ we denote $\bigcup_{Z \subseteq X_t} \widehat{\mathcal{S}}_t[Z]$ (also called a *representative family of partial solutions*). We show how $\widehat{\mathcal{S}}_t$ is obtained by doing dynamic programming from base node to the root node.

*Base node $t$.* Here the graph $H_t$ is empty and thus we take $\widehat{\mathcal{S}}_t = \{\emptyset\}$.

*Introduce node $t$ with child $t'$.* Here, we know that $X_t \supset X_{t'}$ and $|X_t| = |X_{t'}| + 1$. Let $v$ be the vertex in $X_t \setminus X_{t'}$. Furthermore observe that $E(H_t) = E(H_{t'})$ and $v$ is degree zero vertex in $H_t$. Thus the graph $H_t$ only differs from $H_{t'}$ at a isolated vertex $v$. Since we have not added any edge to the new graph, the family of solutions does not change. Thus, we take $\widehat{\mathcal{S}}_t = \widehat{\mathcal{S}}_{t'}$. Formally, we take $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t'}[Z \setminus \{v\}]$. Since, $H_t$ and $H_{t'}$ have same set of edges both the correctness and size invariant is maintained.

*Forget node $t$ with child $t'$.* Here we know $X_t \subset X_{t'}$, $|X_t| = |X_{t'}| - 1$ Let $v \in X_{t'} \setminus X_t$. Observe that $E(H_t) \supseteq E(H_{t'})$. Thus for any $U \in \widehat{\mathcal{S}}_{t'}$, $H_t[U]$ may or may not be a forest. So in this case we collect all the vertex subsets in $\widehat{\mathcal{S}}_{t'}$ which is a forest as induced subgraph in $H_t$. Formally,

$$\widehat{\mathcal{S}}_t[Z] = \left\{ A \in \widehat{\mathcal{S}}_{t'}[Z] \cup \widehat{\mathcal{S}}_{t'}[Z \cup v] \,\middle|\, H_t[A] \text{ is a forest} \right\}.$$

Let $\widehat{\mathcal{S}}_t = \bigcup_{Z \subseteq X_t} \widehat{\mathcal{S}}_t[Z]$. Now we show that $\widehat{\mathcal{S}}_t$ satisfies correctness invariant. Let $L \in \mathscr{S}$. Let $L_{t'} = V(H_{t'}) \cap L$ and $L_R = L \setminus L_{t'}$. Let $Z' = L \cap X_{t'}$ Now observe that

$$\begin{aligned} L \in \mathscr{S} \quad &\Longleftrightarrow \quad L_{t'} \cup L_R \in \mathscr{S} \\ &\Longleftrightarrow \quad \hat{L}_{t'} \cup L_R \in \mathscr{S} \text{ (by the property of } \widehat{\mathcal{S}}_{t'} \text{ we have that } \hat{L}_{t'} \in \widehat{\mathcal{S}}_{t'}[Z']) \end{aligned}$$

Since $H_t[\hat{L}_{t'}]$ is a forest, $\hat{L}_{t'} \in \widehat{\mathcal{S}}_t[Z' \setminus \{v\}]$. This concludes the proof of correctness invariant.

Since $\widehat{\mathcal{S}}_t[Z] \subseteq \widehat{\mathcal{S}}_{t'}[Z] \cup \widehat{\mathcal{S}}_{t'}[Z \cup v]$, the number of partial solutions with $i$ connected components having nonempty intersection with $Z$ in $\widehat{\mathcal{S}}_t[Z]$ is bounded by $\binom{k}{i} + \binom{k+1}{i} \leq \binom{k+2}{i}$. Since $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_t[Z] \otimes_t \{\emptyset\}$, we apply Lemma 6.4 and find $\widehat{\mathcal{S}}'_t[Z] \subseteq \widehat{\mathcal{S}}_t[Z]$ satisfies correctness and size invariant in time $O\left(k^{\omega}\left(2^{\omega}+2\right)^k n + k^{\omega} 2^{k(\omega-1)} 3^k n\right)$ and we set $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}'_t[Z]$.

*Join node $t$ with two children $t_1$ and $t_2$.* Here, we know that $X_t = X_{t_1} = X_{t_2}$. The natural way to get a family of partial solutions for $X_t$ is the union of vertex sets of two families stored at node $t_1$ and $t_2$ which form a forest as an induced subgraph of $H_t$, i.e.,

$$\begin{aligned} \widehat{\mathcal{S}}_t[Z] \quad &= \quad \{U_1 \cup U_2 \mid U_1 \in \widehat{\mathcal{S}}_{t_1}[Z], U_2 \in \widehat{\mathcal{S}}_{t_2}[Z], H_t[U_1 \cup U_2] \text{ is a forest}\} \\ &= \quad \widehat{\mathcal{S}}_{t_1}[Z] \otimes_t \widehat{\mathcal{S}}_{t_2}[Z] \end{aligned}$$

Now we show that $\widehat{\mathcal{S}}_t$ maintains the invariant. Let $L \in \mathscr{S}$. Let $L_t = V(G_t) \cap L, L_{t_1} = V(G_{t_1}) \cap L, L_{t_2} = V(G_{t_2}) \cap L$ and $L_R = L \setminus L_t$. Let $Z = L \cap X_t$ Now observe that

$$\begin{aligned} L \in \mathscr{S} \quad &\Longleftrightarrow \quad L_{t_1} \cup L_{t_2} \cup L_R \in \mathscr{S} \\ &\Longleftrightarrow \quad \hat{L}_{t_1} \cup L_{t_2} \cup L_R \in \mathscr{S} \text{ (by the property of } \widehat{\mathcal{S}}_{t_1} \text{ we have that } \hat{L}_{t_1} \in \widehat{\mathcal{S}}_{t_1}[Z]) \\ &\Longleftrightarrow \quad \hat{L}_{t_1} \cup \hat{L}_{t_2} \cup L_R \in \mathscr{S} \text{ (by the property of } \widehat{\mathcal{S}}_{t_2} \text{ we have that } \hat{L}_{t_2} \in \widehat{\mathcal{S}}_{t_2}[Z]) \end{aligned}$$

We put $\hat{L}_t = \hat{L}_{t_1} \cup \hat{L}_{t_2}$. By the definition of $\widehat{\mathcal{S}}_t[Z]$, we have that $\hat{L}_{t_1} \cup \hat{L}_{t_2} \in \widehat{\mathcal{S}}_t[Z]$. The above inequalities also show that $\hat{L} = \hat{L}_t \cup L_R \in \mathscr{S}$. Note that $(\hat{L}_t \cup L_R) \cap X_t = Z$ This concludes the proof of correctness invariant.

We apply Lemma 6.4 and find $\widehat{\mathcal{S}}'_t[Z] \subseteq \widehat{\mathcal{S}}_t[Z]$ satisfies correctness and size invariant in time $O\left(k^{\omega}\left(2^{\omega}+2\right)^k n + k^{\omega} 2^{k(\omega-1)} 3^k n\right)$ and we set $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}'_t[Z]$.

*Root node $r$.* Here, $X_r = \emptyset$. We go through all the solution in $\widehat{\mathcal{S}}_r[\emptyset]$ and output the one with the maximum weight.

In worst case, in every tree node $t$, for all subset $Z \subseteq X_t$, we apply Lemma 6.4. So by doing the same run time analysis as in the case of Steiner Tree, the total running time will be upper bounded by $O\left(\left(\left(2^{\omega}+3\right)^{\mathbf{tw}} + \left(1 + 2^{\omega-1} \cdot 3\right)^{\mathbf{tw}}\right) \mathbf{tw}^{O(1)} n\right)$. $\square$

## 7 CONCLUSION

In this paper we gave algorithms for finding representative families for product families that are faster than the naive computation for these families. We showed their applicability by designing the best known deterministic algorithms for $k$-wMlD, $k$-wMMlD and for "connectivity problems" parameterized by treewidth. We believe that our algorithms for computing representative families of product families will be useful to accelerate other algorithms. We conclude with several interesting problems.

(1) What are the other natural set families for which we can find representative families faster than by directly applying the results of Fomin et al. [10]?
(2) Can we find representative families for a uniform matroid in time linear in the input size?
(3) Does there exist a deterministic algorithm for $k$-wMLD running in time $2^k n^{O(1)} \log W$?

## REFERENCES


[1] Noga Alon, Raphael Yuster, and Uri Zwick. 1995. Color-coding. *J. Assoc. Comput. Mach.* 42, 4 (1995), 844–856.

[2] Richard Bellman and William Karush. 1962a. Mathematical programming and the maximum transform. *J. Soc. Indust. Appl. Math.* 10 (1962), 550–567.

[3] Richard Bellman and William Karush. 1962b. On the maximum transform and semigroup of transformations. *Bull. Amer. Math. Soc.* 68 (1962), 516–518.

[4] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. 2007. Fourier meets Möbious: Fast subset convolution. In *Proceedings of the 39th annual ACM Symposium on Theory of Computing (STOC 2007)*. ACM Press, New York, to appear.

[5] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. 2010. Narrow sieves for parameterized paths and packings. *CoRR* abs/1007.1161 (2010).

[6] Andreas Björklund, Petteri Kaski, and Lukasz Kowalik. 2013. Probably Optimal Graph Motifs. In *STACS (LIPIcs)*, Vol. 20. 20–31.

[7] Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. 2013. Solving weighted and counting variants of connectivity problems parameterized by treewidth deterministically in single exponential time, In ICALP. *CoRR* (2013), 196–207.

[8] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer. DOI:http://dx.doi.org/10.1007/978-3-319-21275-3

[9] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. 2011. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS 2011)*. IEEE.

[10] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. 2016. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM* 63, 4 (2016), 29:1–29:60. DOI:http://dx.doi.org/10.1145/2886094

[11] Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh, and B. V. Raghavendra Rao. 2012. Faster algorithms for finding and counting subgraphs. *J. Comput. System Sci.* 78, 3 (2012), 698–706. DOI:http://dx.doi.org/10.1016/j.jcss.2011.10.001

[12] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. 2014. Efficient Computation of Representative Sets with Applications in Parameterized and Exact Algorithms. In *SODA*. 142–151.

[13] François Le Gall. 2014. Powers of Tensors and Fast Matrix Multiplication. *CoRR* abs/1401.7714 (2014). http://arxiv.org/abs/1401.7714

[14] Sylvain Guillemot and Florian Sikora. 2013. Finding and Counting Vertex-Colored Subtrees. *Algorithmica* 65, 4 (2013), 828–844.

[15] Stasys Jukna. 2011. *Extremal combinatorics*. Springer Verlag Berlin Heidelberg.

[16] Ton Kloks. 1994. *Treewidth, Computations and Approximations*. Lecture Notes in Computer Science, Vol. 842. Springer.

[17] Ioannis Koutis. 2008. Faster Algebraic Algorithms for Path and Packing Problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008) (Lecture Notes in Computer Science)*, Vol. 5125. 575–586.

[18] Ioannis Koutis. 2012. Constrained multilinear detection for faster functional motif discovery. *Inf. Process. Lett.* 112, 22 (2012), 889–892.

[19] Ioannis Koutis and Ryan Williams. 2009. Limits and Applications of Group Algebras for Parameterized Problems. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP 2009) (Lecture Notes in Computer Sci.)*, Vol. 5555. Springer, 653–664.

[20] Ioannis Koutis and Ryan Williams. 2016. Algebraic fingerprints for faster algorithms. *Commun. ACM* 59, 1 (2016), 98–105. DOI:http://dx.doi.org/10.1145/2742544

[21] Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. 2015. Deterministic Truncation of Linear Matroids. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*. 922–934. DOI:http://dx.doi.org/10.1007/978-3-662-47672-7_75

[22] László Lovász. 1977. Flats in matroids and geometric graphs.. In *In Combinatorial surveys (Proc. Sixth British Combinatorial Conf., Royal Holloway Coll., Egham)*. Academic Press, London, 45–86.

[23] Dániel Marx. 2006. Parameterized coloring problems on chordal graphs. *Theor. Comput. Sci.* 351, 3 (2006), 407–424.

[24] Dániel Marx. 2009. A parameterized view on matroid optimization problems. *Theor. Comput. Sci.* 410, 44 (2009), 4471–4479.

[25] Burkhard Monien. 1985. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*. North-Holland Math. Stud., Vol. 109. North-Holland, Amsterdam, 239–254. DOI: http://dx.doi.org/10.1016/S0304-0208(08)73110-4

[26] James G Oxley. 2006. *Matroid theory*. Vol. 3. Oxford University Press.

[27] Fahad Panolan and Meirav Zehavi. 2016. Parameterized Algorithms for List K-Cycle. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15, 2016, Chennai, India*. 22:1–22:15. http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2016.22

[28] Zsolt Tuza. 1994. Applications of the set-pair method in extremal hypergraph theory. In *Extremal problems for finite sets (Visegrád, 1991)*. Bolyai Soc. Math. Stud., Vol. 3. János Bolyai Math. Soc., Budapest, 479–514.

[29] Zsolt Tuza. 1996. Applications of the set-pair method in extremal problems. II. In *Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993)*. Bolyai Soc. Math. Stud., Vol. 2. János Bolyai Math. Soc., Budapest, 459–490.

[30] Ryan Williams. 2009. Finding Paths of Length $k$ in $O^*(2^k)$ Time. *Inf. Process. Lett.* 109, 6 (2009), 315–318.

[31] Virginia Vassilevska Williams. 2012. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC 2012)*. ACM, 887–898.


Received February 2007; revised March 2009; accepted June 2009