Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms

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

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 $\widehat{X} \cup Y \in \mathcal{I}$. By the classic result of Bollobás, in a uniform matroid, every family of sets of size p has a q-representative family with at most $\binom{p+q}{p}$ sets. In his famous "two families theorem" from 1977, Lovász proved that the same bound also holds for any matroid representable over a field $\mathbb{F}$. We give an efficient construction of a q-representative family of size at most $\binom{p+q}{p}$ in time bounded by a polynomial in $\binom{p+q}{p}$, t, and the time required for field operations.

We demonstrate how the efficient construction of representative families can be a powerful tool for designing single-exponential parameterized and exact exponential time algorithms. The applications of our approach include the following:

—In the Long Directed Cycle problem, the input is a directed n-vertex graph G and the positive integer k. The task is to find a directed cycle of length at least k in G, if such a cycle exists. As a consequence of our $6.75^{k+o(k)}n^{\mathcal{O}(1)}$ time algorithm, we have that a directed cycle of length at least $\log n$, if such a cycle exists, can be found in polynomial time.
—In the Minimum Equivalent Graph (MEG) problem, we are seeking a spanning subdigraph D' of a given n-vertex digraph D with as few arcs as possible in which the reachability relation is the same as in the original digraph D.
—We provide an alternative proof of the recent results for algorithms on graphs of bounded treewidth showing that many "connectivity" problems such as Hamiltonian Cycle or Steiner Tree can be solved in time $2^{\mathcal{O}(t)}n$ on n-vertex graphs of treewidth at most t.

For the special case of uniform matroids on n elements, we give a faster algorithm to compute a representative family. We use this algorithm to provide the fastest known deterministic parameterized algorithms for k-Path, k-Tree, and, more generally, k-Subgraph Isomorphism, where the k-vertex pattern graph is of constant treewidth.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.2.2 [**Graph Theory**]: Graph Algorithms

General Terms: Algorithms, Design, Theory

Preliminary versions of this article appeared in the proceedings of SODA 2014 and ESA 2014. This work is supported by Rigorous Theory of Preprocessing, ERC Advanced Investigator Grant 267959, and Parameterized Approximation, ERC Starting Grant 306992.
Authors' addresses: F. V. Fomin and D. Lokshtanov, Department of Informatics, University of Bergen, Postboks 7803 5020 Bergen, Norway; emails: {fomin, daniello}@ii.uib.no; F. Panolan and S. Saurabh, Theoretical Computer Science, Institute of Mathematical Sciences, Chennai, India; emails: fahad.panolan@gmail.com, saket@imsc.res.in.

DOI: http://dx.doi.org/10.1145/2886094

Additional Key Words and Phrases: Matroids, representative families, linear independence, hash functions, parameterized algorithms

ACM Reference Format:
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, Article 29 (September 2016), 60 pages.
DOI: http://dx.doi.org/10.1145/2886094

1. INTRODUCTION

The theory of matroids provides deep insight into the tractability of many fundamental problems in Combinatorial Optimization like Minimum Weight Spanning Tree or Perfect Matching. Marx [2009] was the first to apply matroids to design fixed-parameter tractable algorithms. The main tool used by Marx was the notion of *representative families*. Representative families for set systems were introduced by Monien [1985].

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 and $\widehat{X} \cup Y \in \mathcal{I}$. In other words, if a set Y of size at most q can be extended to an independent set of size $|Y| + p$ by adding a subset from $\mathcal{S}$, then it also can be extended to an independent set of size $|Y| + p$ by adding a subset from $\widehat{\mathcal{S}}$ as well.

The Two-Families Theorem of Bollobás [1965] for extremal set systems and its generalization to subspaces of a vector space of Lovász [1977] (see also Frankl [1982]) imply that every family of sets of size p has a q-representative family with at most $\binom{p+q}{p}$ sets. These theorems are the cornerstones in extremal set theory with numerous applications in graph and hypergraph theory, combinatorial geometry, and theoretical computer science. We refer to Section 9.2.2 of Jukna [2011], surveys of Tuza [1994, 1996], and Gil Kalai's blog[1] for more information on the theorems and their applications.

For set families, or equivalently for uniform matroids, Monien provided an algorithm computing a q-representative family of size at most $\sum_{i=0}^{q} p^i$ in time $\mathcal{O}(pq \cdot \sum_{i=0}^{q} p^i \cdot t)$ [Monien 1985]. Marx [2006] provided another algorithm, also for uniform matroids, for finding q-representative families of size at most $\binom{p+q}{p}$ in time $\mathcal{O}(p^q \cdot t^2)$. For linear matroids, Marx [2009] has shown how Lovász's proof can be transformed into an algorithm computing a q-representative family. However, the running time of the algorithm given in Marx [2009] is $f(p,q)(||A_M||t)^{\mathcal{O}(1)}$, where $f(p,q)$ is a polynomial in $(p+q)^p$ and $\binom{p+q}{p}$, that is, $f(p,q) = 2^{\mathcal{O}(p \log(p+q))} \cdot \binom{p+q}{p}^{\mathcal{O}(1)}$, and A_M is the matroid's representation matrix. Thus, when p is a constant, which is the way this lemma has been recently used in the kernelization algorithms [Kratsch and Wahlström 2012], we have that $f(p,q) = (p+q)^{\mathcal{O}(1)}$. However, for unbounded p (for an example when $p = q = \frac{k}{2}$), the running time of this algorithm is bounded by $2^{\mathcal{O}(k \log k)}(||A_M||t)^{\mathcal{O}(1)}$.

Our results. We give two faster algorithms computing representative families and show how they can be used to obtain improved parameterized and exact exponential algorithms for several fundamental and well-studied problems.

Our first result is the following.

Theorem 1.1. *Let $M = (E, \mathcal{I})$ be a linear matroid of rank $p + q = k$ given together with its representation matrix A_M over a field $\mathbb{F}$. Let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a family of independent sets of size p. Then a q-representative family $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ for $\mathcal{S}$ with at most $\binom{p+q}{p}$*

[1] http://gilkalai.wordpress.com/2008/12/25/lovaszs-two-families-theorem/.

sets can be found in $\mathcal{O}(\binom{p+q}{p}tp^{\omega} + t\binom{p+q}{q}^{\omega-1})$ *operations over* $\mathbb{F}$. *Here,* $\omega < 2.373$ *is the matrix multiplication exponent.*

Actually, we will prove a variant of Theorem 1.1 that allows sets to have weights. This extension will be used in several applications. This theorem uses the notion of weighted representative families and computes a weighted q-representative family of size at most $\binom{p+q}{p}$ within the running time claimed in Theorem 1.1. The proof of Theorem 1.1 relies on the exterior algebra-based proof of Lovász [1977] and exploits the multilinearity of the determinant function.

For the case of uniform matroids, we provide the following theorem.

THEOREM 1.2. *Let* $\mathcal{S} = \{S_1, \ldots, S_t\}$ *be a family of sets of size* p *over a universe of size* n *and let* $0 < x < 1$. *For a given* q, *a* q*-representative family* $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ *for* $\mathcal{S}$ *with at most* $x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)}$ *sets can be computed in time* $\mathcal{O}((1-x)^{-q} \cdot 2^{o(p+q)} \cdot t \cdot \log n)$.

As in the case of Theorem 1.1, we prove a more general version of Theorem 1.2 for weighted sets. The proof of Theorem 1.2 is essentially an algorithmic variant of the "random permutation" proof of the Bollobás Lemma (see Jukna [2011, Theorem 8.7]). A slightly weaker variant of the Bollobás Lemma can be proved using random partitions instead of random permutations, the advantage of the random partitions proof being that it can be de-randomized using efficient constructions of *universal sets* [Naor et al. 1995]. To obtain our results, we define *separating collections* and give efficient constructions of them.

Separating collections can be seen as a variant of universal sets. In its simplest form, an *n-p-q-separating collection* $\mathcal{C}$ is a pair $(\mathcal{F}, \chi)$, where $\mathcal{F}$ is a family of sets over a universe U of size n and χ is a function from $\binom{U}{p}$ to $2^{\mathcal{F}}$ such that the following two properties are satisfied: (1) for every $A \in \binom{U}{p}$ and every $F \in \chi(A)$, $A \subseteq F$, and (2) for every $A \in \binom{U}{p}$ and $B \in \binom{U \setminus A}{q}$, there is an $F \in \chi(A)$ such that $A \subseteq F$ and $F \cap B = \emptyset$. The *size* of $(\mathcal{F}, \chi)$ is $|\mathcal{F}|$, whereas the *max degree* of $(\mathcal{F}, \chi)$ is $\max_{A \in \binom{U}{p}} |\chi(A)|$. Here 2^S for a set S is the family of all subsets of S, while $\binom{S}{p}$ is the family of all subsets of S of size p.

An efficient construction of separating collections is an algorithm that given n, p, and q, outputs the family $\mathcal{F}$ of a separating collection $(\mathcal{F}, \chi)$ and then allows queries $\chi(A)$ for $A \in \binom{U}{p}$. We give constructions of separating collections of optimal (up to subexponential factors in $p+q$) size and degree, and construction and query time that is linear (up to subexponential factors in $p+q$) in the size of the output.

In the conference version of the article [Fomin et al. 2014a], we only proved Theorem 1.2 for $x = \frac{p}{p+q}$. That is, let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a family of sets of size p over a universe of size n. Then, for a given q, a q-representative family $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ for $\mathcal{S}$ with at most $\binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ sets can be computed in time $\mathcal{O}(\binom{p+q}{q}^{q} \cdot 2^{o(p+q)} \cdot t \cdot \log n)$. Later we observed that our proof works for every $0 < x < 1$ and allows an interesting tradeoff between the size of the computed representative families and the time taken to compute them [Fomin et al. 2014b], and that this tradeoff can be exploited algorithmically to speed up "representative-families-based" algorithms. Theorem 1.2 improves the one in Fomin et al. [2014a] by shaving off a multiplicative factor of $\log n$ from the upper bound on the output family size. Independently, at the same time, Shachnai and Zehavi [2014b] also observed that our initial proof could be generalized in essentially the same way as what is stated in Theorem 1.2, and that this generalization used to speed up some of the algorithms given in the preliminary version of the article [Fomin

Table I. We Use $\mathcal{O}^*()$ Notation that Hides Factors Polynomial in the Number of Vertices n and the Parameter k in Cases When the Authors Do Not Specify the Power of Polynomials

Reference	Randomized	Deterministic
Monien [1985]	-	$\mathcal{O}(k!nm)$
Bodlaender [1993]	-	$\mathcal{O}(k!2^k n)$
Alon et al. [1995]	$\mathcal{O}(5.44^k n)$	$\mathcal{O}(c^k n \log n)$ for a large c
Hüffner et al. [2008]	$\mathcal{O}(4.32^k m)$	
Kneis et al. [2008]	$\mathcal{O}^*(4^k)$	$\mathcal{O}^*(16^k)$
Chen et al. [2009]	$\mathcal{O}(4^k k^{2.7} m)$	$4^{k+\mathcal{O}(\log^3 k)}nm$
Koutis [2008]	$\mathcal{O}^*(2.83^k)$	-
Williams [2009]	$\mathcal{O}^*(2^k)$	-
Björklund et al. [2010]	$\mathcal{O}^*(1.66^k)$	-
Conference version	-	$\mathcal{O}(2.851^k n \log^2 n)$
This article	-	$\mathcal{O}(2.619^k n \log n)$

et al. 2014a]. In particular, they obtain the same dependence on k in the running time bounds as in this article for k-Path and Long Directed Cycle.

Applications. Here we provide the list of main applications that can be derived from our algorithms that compute representative families together with a short overview of previous work on each application.

***k*-Path.** In the k-Path problem, we are given an undirected n-vertex graph G and integer k. The question is if G contains a path of length k. k-Path was studied intensively within the parameterized complexity paradigm [Downey and Fellows 1999]. For n-vertex graphs, the problem is trivially solvable in time $\mathcal{O}(n^k)$. Monien [1985] and Bodlaender [1993] showed that the problem is fixed-parameter tractable. Monien [1985] used representative families for set systems for his k-Path algorithm, and Plehn and Voigt [1991] extended this algorithm to Subgraph Isomorphism. This led Papadimitriou and Yannakakis [1996] to conjecture that the problem is solvable in polynomial time for $k = \log n$. This conjecture was resolved in a seminal paper of Alon et al. [1995], who introduced the method of color-coding and obtained the first single exponential algorithm for the problem. Actually, the method of Alon et al. can be applied for more general problems, like finding a k-path in directed graphs, or to solve the Subgraph Isomorphism problem in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(t)}$, when the treewidth of the pattern graph is bounded by t. There has been a lot of effort in parameterized algorithms to reduce the base of the exponent of both deterministic and the randomized algorithms for the k-Path problem (see Table I). After the work of Alon et al. [1995], there were several breakthrough ideas leading to faster and faster *randomized* algorithms. We refer to Fomin and Kaski [2013], Koutis and Williams [2016], and Cygan et al. [2015, Chapter 10] for an extensive overview of parameterized randomized algorithms for k-Path. Concerning deterministic algorithms, no improvements occurred since 2007, when Chen et al. [2007] showed a clever way of applying universal sets to reduce the running time of color-coding the algorithm to $\mathcal{O}^*(4^{k+o(k)})$.

k-Path is a special case of the k-Subgraph Isomorphism problem, where for given n-vertex graph G and k-vertex graph F, the question is whether G contains a subgraph isomorphic to F. In addition to k-Path, parameterized algorithms for two other variants of k-Subgraph Isomorphism, when F is a tree, and more generally, a graph of treewidth at most t, were studied in the literature. Alon et al. [1995] showed that k-Subgraph Isomorphism, when the treewidth of the pattern graph is bounded by t, is solvable in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(t)}$. Cohen et al. [2010] gave a randomized algorithm that for an input digraph D decides in time $5.704^k n^{\mathcal{O}(1)}$ if D contains a given out-tree with k vertices. They

also showed how to de-randomize the algorithm in time $6.14^k n^{\mathcal{O}(1)}$. Amini et al. [2012] introduced an inclusion-exclusion-based approach in the classic color-coding and gave a randomized $5.4^k n^{\mathcal{O}(t)}$ time algorithm and a deterministic $5.4^{k+o(k)} n^{\mathcal{O}(t)}$ time algorithm for the case when F has treewidth at most t. Koutis and Williams [2009] generalized their algebraic approach for k-Path to k-Tree and obtained a randomized algorithm running in time $2^k n^{\mathcal{O}(1)}$ for k-Tree. Fomin et al. [2012] extended this result by providing a randomized algorithm for k-Subgraph Isomorphism running in time $2^k (nt)^{\mathcal{O}(t)}$, when the treewidth of F is at most t. However, the fastest known deterministic algorithm for this problem prior to this article was the time $5.4^{k+o(k)} n^{\mathcal{O}(t)}$ algorithm from Amini et al. [2012]. In this article, we give deterministic algorithms for k-Path and k-Tree that run in time $\mathcal{O}(2.619^k n \log n)$ and $\mathcal{O}(2.619^k n^{\mathcal{O}(1)})$. The algorithm for k-Tree can be generalized to k-Subgraph Isomorphism for the case when the pattern graph F has treewidth at most t. This algorithm will run in time $\mathcal{O}(2.619^k n^{\mathcal{O}(t)})$. Our approach can also be applied to find directed paths and cycles of length k in time $\mathcal{O}(2.619^k m \log n)$ and $\mathcal{O}(2.619^k n^{\mathcal{O}(1)})$, respectively.

Another interesting feature of our approach is that due to using weighted representative families, we can handle the weighted version of the problem as well. The weighted version of k-Path is known as Short Cheap Tour. Let G be a graph with maximum edge cost W, and then the problem is to find a path of length at least k where the total sum of costs on the edges is minimized. The algorithm of Björklund et al. [2010] can be adapted to solve Short Cheap Tour in time $\mathcal{O}(1.66^k n^{\mathcal{O}(1)} W)$; however, their approach does not seem to be applicable to obtain algorithms with polylogarithmic dependence on W. Williams [2009] observed that a divide-and-color approach from Chen et al. [2009] can be used to solve Short Cheap Tour in time $\mathcal{O}(4^k n^{\mathcal{O}(1)} \log W)$. No better algorithm for Short Cheap Tour was known prior to our work. As it was noted by Williams, the $\mathcal{O}(2^k n^{\mathcal{O}(1)})$ algorithm of his paper does not appear to extend to weighted graphs. Our approach provides a deterministic $\mathcal{O}(2.619^k n^{\mathcal{O}(1)} \log W)$ time algorithm for Short Cheap Tour and partially resolves an open question asked by Williams.

Long Directed Cycle. In the Long Directed Cycle problem, we are interested in finding a cycle of length at least k in a directed graph. For this problem, we give an algorithm of running time $\mathcal{O}(6.75^{k+o(k)} m n^2 \log n)$.

While at first glance the problem is similar to the problem of finding a cycle or a path of length exactly k, it is more tricky. The reason is that the problem of finding a cycle of length $\geq k$ may entail finding a much longer, potentially even a Hamiltonian cycle. This is why color-coding and other techniques applicable to k-Path do not seem to work here. Even for undirected graphs, color-coding alone is not sufficient, and one needs an additional clever trick to make it work. The first fixed-parameter tractable algorithm for Long Directed Cycle is due to Gabow and Nie [2008], who gave algorithms with expected running time $k^{2k} 2^{\mathcal{O}(k)} nm$ and worst-case times $\mathcal{O}(k^{2k} 2^{\mathcal{O}(k)} nm \log n)$ or $\mathcal{O}(k^{3k} nm)$. These running times allow them to find a directed cycle of length at least $\log n / \log \log n$ in expected polynomial time, if it exists. Let us note that our algorithm implies that one can find in polynomial time a directed cycle of length at least $\log n$ if there is such a cycle. On the other hand, Björklund et al. [2004] have shown that assuming the Exponential Time Hypothesis (ETH) of Impagliazzo et al. [2001], there is no polynomial time algorithm that finds a directed cycle of length $\Omega(f(n) \log n)$, for any nondecreasing, unbounded, polynomial time computable function f that tends to infinity. Thus, our work closes the gap between the upper and lower bounds for this problem.

Minimum Equivalent Graph. Our next application is from exact exponential time algorithms; we refer to Fomin and Kratsch [2011] for an introduction to the area of

exact algorithms. In the MINIMUM EQUIVALENT GRAPH (MEG) problem, we are seeking a spanning subdigraph D' of a given digraph D with as few arcs as possible in which the reachability relation is the same as in the original digraph D. In other words, for every pair of vertices u, v, there is a path from u to v in D' if and only if the original digraph D has such a path. We show that this problem is solvable in time $\mathcal{O}(2^{4\omega n}mn)$, where n is the number of vertices and m is the number of arcs in D.

MEG is a classic NP-hard problem generalizing the HAMILTONIAN CYCLE problem; see Chapter 12 of the book by Bang-Jensen and Gutin [2009] for an overview of combinatorial and algorithmic results on MEG. The algorithmic studies of MEG can be traced to the work of Moyles and Thompson [1969] from 1969, who gave a (nontrivial) branching algorithm solving MEG in time $\mathcal{O}(n!)$. In 1975, Hsu [1975] discovered a mistake in the algorithm of Moyles and Thompson and designed a different branching algorithm for this problem. Martello [1978] and Martello and Toth [1982] gave another branching-based algorithm with running time $\mathcal{O}(2^m)$. No single-exponential exact algorithm, that is, of running time $2^{\mathcal{O}(n)}$, for MEG was known prior to our work.

As it was already observed by Moyles and Thompson [1969], the hardest instances of MEG are strong digraphs. A digraph is strong if for every pair of vertices $u \neq v$, there are directed paths from u to v and from v to u. MEG restricted to strong digraphs is known as the MINIMUM SCSS (strongly connected spanning subgraph) problem. It is known that the MEG problem reduces in linear time to MINIMUM SCSS (see, e.g., Cormen et al. [2001]).

Treewidth Algorithms. We show that efficient computation of representative families can be used to obtain algorithms solving "connectivity" problems like HAMILTONIAN CYCLE or STEINER TREE in time $2^{\mathcal{O}(t)}n$, where t is the treewidth of the input n-vertex graph. 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 $2^{\mathcal{O}(t)}n$. 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. [2011] introduced a new algorithmic framework called Cut&Count and used it to obtain $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$ time Monte Carlo algorithms for a number of connectivity problems. Recently, Bodlaender et al. [2015] obtained the first deterministic single exponential algorithms for these problems. Bodlaender et al. presented two approaches, one based on rank estimations in specific matrices and the second based on matrix-tree theorem and computation of determinants. Our approach, based on representative families in matroids, can be seen as an alternate path to obtain similar results. The main idea behind our approach is that all the relevant information about "partial solutions" in bags of the tree decomposition can be encoded as an independent set of a specific matroid. Here efficient computation of representative families comes into play.

In all our applications, we first define a specific matroid and then show a combinatorial relation between solutions to the problem and independent sets of the matroid. Then we compute representative families using Theorem 1.1 or Theorem 1.2 and use them to obtain a solution to the problem. We believe that expressing graph problems in "matroid language" is a generic technique explaining why certain problems admit single-exponential parameterized and exact exponential algorithms. Finally, for completeness, we would like to add that in the conference version of the article, the running time for k-PATH and k-TREE were $\mathcal{O}(2.815^k n^{\mathcal{O}(1)})$, for k-SUBGRAPH ISOMORPHISM for the case when the pattern graph F has treewidth at most t was $\mathcal{O}(2.815^k n^{\mathcal{O}(t)})$, and for LONG DIRECTED CYCLE was $8^{k+o(k)}n^{\mathcal{O}(1)}$.

Organization of the Paper. In Section 2, we give the necessary definitions and state some of the known results that we will use. In Section 3, we prove Theorem 1.1 by giving an efficient algorithm for the computation of representative families for linear matroids. In Section 4, we prove Theorem 1.2 by giving an efficient algorithm for the computation of representative families for uniform matroids. In Section 5, we give all our applications of Theorems 1.1 and 1.2. Concluding remarks and new developments can be found in Section 6. The proofs of Theorem 1.1 and Theorem 1.2 are independent of each other and may be read independently. All of our applications use Theorems 1.1 and 1.2 as black boxes, and thus may be read independently of the sections describing the efficient computation of representative families.

2. PRELIMINARIES

In this section, we give various definitions that we make use of in the article.

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 $G[V(G) \setminus S]$. By $N(u)$ we denote (open) neighborhood of u, that is, the set of all vertices adjacent to u. Similarly, by $N[u] = N(u) \cup \{u\}$ we define the closed neighborhood. The degree of a vertex v in G is $|N_G(v)|$ and is denoted by $d(v)$. For a subset $S \subseteq V(G)$, we define $N[S] = \cup_{v \in S} N[v]$ and $N(S) = N[S] \setminus S$. By the length of the path we mean the number of edges in it.

Digraphs. Let D be a digraph. By $V(D)$ and $A(D)$ we represent the vertex set and arc set of D, respectively. Given a subset $V' \subseteq V(D)$ of a digraph D, let $D[V']$ denote the digraph induced by V'. A digraph D is *strong* if for every pair x, y of vertices there are directed paths from x to y and from y to x. A maximal strongly connected subdigraph of D is called a *strong component*. A vertex u of D is an *in-neighbor* (*out-neighbor*) of a vertex v if $uv \in A(D)$ ($vu \in A(D)$, respectively). The *in-degree* $d^-(v)$ (*out-degreed*$^+(v)$) of a vertex v is the number of its in-neighbors (out-neighbors). We denote the set of in-neighbors and out-neighbors of a vertex v by $N^-(v)$ and $N^+(v)$ correspondingly. A *closed directed walk* in a digraph D is a sequence $v_0 v_1 \cdots v_\ell$ of vertices of D, not necessarily distinct, such that $v_0 = v_\ell$ and for every $0 \leq i \leq \ell - 1$, $v_i v_{i+1} \in A(D)$.

Sets, Functions, and Constants. We use the following notations: $[n] = \{1, \ldots, n\}$ and $\binom{[n]}{i} = \{X \mid X \subseteq [n],\ |X| = i\}$.

We use the following operations on families of sets.

Definition 2.1. Given two families of sets $\mathcal{A}$ and $\mathcal{B}$, we define

(•) $\mathcal{A} \bullet \mathcal{B} = \{X \cup Y \mid X \in \mathcal{A} \text{ and } Y \in \mathcal{B} \text{ and } X \cap Y = \emptyset\}$. Let $\mathcal{A}_1, \ldots, \mathcal{A}_r$ be r families. Then

$$\prod_{i \in [r]}^{\bullet} \mathcal{A}_i = \mathcal{A}_1 \bullet \cdots \bullet \mathcal{A}_r.$$

(○) $\mathcal{A} \circ \mathcal{B} = \{A \cup B : A \in \mathcal{A} \text{ and } B \in \mathcal{B}\}$.

(+) For a set X, we define $\mathcal{A} + X = \{A \cup X : A \in \mathcal{A}\}$.

The first and second derivatives of a function $f(x)$ of a variable x are denoted by $f'(x)$ and $f''(x)$, respectively. Throughout the article, we use ω to denote the exponent in the running time of matrix multiplication, the current best-known bound for which is $\omega < 2.373$ [Williams 2012]. We use e to denote the base of natural logarithm.

2.1. Randomized Algorithms

We follow the same notion of randomized algorithms as described in Marx [2009, Section 2.3]. That is, some of the algorithms presented in this article are randomized, which means that they can produce an incorrect answer, but the probability of doing so is small. We assume that the algorithm has an integer parameter P given in unary, and the probability of incorrect answer is 2^{-P}.

2.2. Matroids

In the next few subsections, we give definitions related to matroids. For a broader overview on matroids, we refer to Oxley [2006]; see also Cygan et al. [2015, Chapter 12].

Definition 2.2. 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) $\phi \in \mathcal{I}$.
(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 is $e \in (B \setminus A)$ 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 the 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 $\mathsf{rank}(M)$.

2.3. 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. For A, we define matroid $M = (E, \mathcal{I})$ as follows. A set $X \subseteq E$ is independent (i.e., $X \in \mathcal{I}$) 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 $\mathbb{F}^d$ corresponding 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 $\mathbb{F}$.

2.4. Direct Sum of Matroids

Let $M_1 = (E_1, \mathcal{I}_1)$, $M_2 = (E_2, \mathcal{I}_2), \ldots, M_t = (E_t, \mathcal{I}_t)$ be t matroids with $E_i \cap E_j = \emptyset$ for all $1 \leq i \neq j \leq t$. The direct sum $M_1 \oplus \cdots \oplus M_t$ is a matroid $M = (E, \mathcal{I})$ with $E := \bigcup_{i=1}^{t} E_i$, and $X \subseteq E$ is independent if and only if $X \cap E_i \in \mathcal{I}_i$ for all $i \leq t$. Let A_i be the representation matrix of $M_i = (E_i, \mathcal{I}_i)$. Then,

$$A_M = \begin{pmatrix} A_1 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_t \end{pmatrix}$$

is a representation matrix of $M_1 \oplus \cdots \oplus M_t$. The correctness of this construction is proved in Marx [2009].

Proposition 2.3 [Marx 2009, Proposition 3.4]. *Given representations of matroids $M_1, \ldots, M_t$ over the same field $\mathbb{F}$, a representation of their direct sum can be found in polynomial time.*

2.5. Uniform and Partition 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 $U_{n,k}$. Every uniform matroid is linear and can be represented over a finite field by a $k \times n$ matrix A_M the $A_M[i, j] = j^{i-1}$:

$$A_M = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ 1 & 2^2 & 3^2 & \cdots & n^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 2^{k-1} & 3^{k-1} & \cdots & n^{k-1} \end{pmatrix}.$$

Matrix A_M is called a Vandermonde matrix. Observe that for $U_{n,k}$ to be representable over a finite field $\mathbb{F}$, the determinant of each $k \times k$ submatrix of A_M must not vanish over $\mathbb{F}$. Observe that any k columns corresponding to $x_{i_1}, \ldots, x_{i_k}$ itself form a Vandermonde matrix, whose determinant is given by

$$\prod_{1 \leq j < \ell \leq k} (x_{i_j} - x_{i_\ell}).$$

Combining this with the fact that $x_1, \ldots, x_n$ are n distinct elements of $\mathbb{F}$, we conclude that every subset of size at most k of the ground set is independent, while clearly each larger subset is dependent. Thus, choosing a field $\mathbb{F}$ of size larger than n suffices. Note that this means that a representation of the uniform matroid $U_{n,k}$ can be stored using $\mathcal{O}(\log n)$ bits.

A partition matroid $M = (E, \mathcal{I})$ is defined by a ground set E being partitioned into (disjoint) sets $E_1, \ldots, E_\ell$ and by ℓ nonnegative integers $k_1, \ldots, k_\ell$. A set $X \subseteq E$ is independent if and only if $|X \cap E_i| \leq k_i$ for all $i \in \{1, \ldots, \ell\}$. Observe that a partition matroid is a direct sum of uniform matroids $U_{|E_1|,k_1}, \ldots, U_{|E_\ell|,k_\ell}$. Thus, by Proposition 2.3 and the fact that a uniform matroid $U_{n,k}$ is representable over a field $\mathbb{F}$ of size larger than n, we have the following.

Proposition 2.4 [Marx 2009, Proposition 3.5]. *A representation over a field of size $\mathcal{O}(|E|)$ of a partition matroid can be constructed in polynomial time.*

2.6. 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 forest in the graph G. We can get a representation of graphic matroid as follows. 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 simply needs to take the representation given above over reals and simply replace all -1 by the additive inverse of 1.

Proposition 2.5 [Oxley 2006]. *Graphic matroids are representable over any field of size at least 2.*

2.7. Truncation of a Matroid

The *t-truncation* of a matroid $M = (E, \mathcal{I})$ is a matroid $M' = (E, \mathcal{I}')$ such that $S \subseteq E$ is independent in M' if and only if $|S| \leq t$ and S is independent in M (i.e., $S \in \mathcal{I}$).

PROPOSITION 2.6 [MARX 2009, PROPOSITION 3.7]. *Given a matroid M with a representation A over a finite field $\mathbb{F}$ and an integer t, a representation of the t-truncation M' can be found in randomized polynomial time.*

3. FAST COMPUTATION FOR REPRESENTATIVE FAMILIES FOR LINEAR MATROIDS

In this section, we give an algorithm to find a q-representative family of a given family. We start with the definition of a *q-representative family*.

Definition 3.1 (*q-Representative Family*). 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 $\hat{\mathcal{S}} \subseteq \mathcal{S}$ 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 adding 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 3.2 (*Min/Max q-Representative Family*). Given a matroid $M = (E, \mathcal{I})$, a family $\mathcal{S}$ of subsets of E, and a nonnegative 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 $\mathcal{S}$.

We say that a family $\mathcal{S} = \{S_1, \ldots, S_t\}$ of sets is a *p-family* if each set in $\mathcal{S}$ is of size p.

We start with three lemmata providing basic results about representative families. These lemmata will be used in Section 5, where we provide algorithmic applications of representative families. We prove them for unweighted representative families, but they can be easily modified to work for weighted variant.

LEMMA 3.3. *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 $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$.*

PROOF. Let $Y \subseteq E$ of size at most q such that there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$. By the definition of q-representative family, we have that there is a set $X' \in \mathcal{S}'$ disjoint from Y with $X' \cup Y \in \mathcal{I}$. Now the fact that $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}'$ yields that there exists a $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y with $\widehat{X} \cup Y \in \mathcal{I}$. □

LEMMA 3.4. *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 $\cup_{i=1}^{\ell} \widehat{\mathcal{S}}_i \subseteq^q_{rep} \mathcal{S}$.*

PROOF. Let $Y \subseteq E$ of size at most q such that there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$. Since $\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_\ell$, there exists an i such that $X \in \mathcal{S}_i$. This implies that there exists a $\widehat{X} \in \widehat{\mathcal{S}}_i \subseteq \cup_{i=1}^{\ell} \widehat{\mathcal{S}}_i$ disjoint from Y with $\widehat{X} \cup Y \in \mathcal{I}$. □

LEMMA 3.5. *Let $M = (E, \mathcal{I})$ be a matroid of rank k and $\mathcal{S}_1$ be a p_1-family of independent sets, $\mathcal{S}_2$ be a p_2-family of independent sets, $\widehat{\mathcal{S}}_1 \subseteq_{rep}^{k-p_1} \mathcal{S}_1$, and $\widehat{\mathcal{S}}_2 \subseteq_{rep}^{k-p_2} \mathcal{S}_2$. Then $\widehat{\mathcal{S}}_1 \bullet \widehat{\mathcal{S}}_2 \subseteq_{rep}^{k-p_1-p_2} \mathcal{S}_1 \bullet \mathcal{S}_2$.*

PROOF. Let $Y \subseteq E$ of size at most $q = k - p_1 - p_2$ such that there is a set $X \in \mathcal{S}_1 \bullet \mathcal{S}_2$ disjoint from Y with $X \cup Y \in \mathcal{I}$. This implies that there exist $X_1 \in \mathcal{S}_1$ and $X_2 \in \mathcal{S}_2$ such that $X_1 \cup X_2 = X$ and $X_1 \cap X_2 = \emptyset$. Since $\widehat{\mathcal{S}}_1 \subseteq_{rep}^{k-p_1} \mathcal{S}_1$, we have that there exists a $\widehat{X}_1 \in \widehat{\mathcal{S}}_1$ such that $\widehat{X}_1 \cup X_2 \cup Y \in \mathcal{I}$ and $\widehat{X}_1 \cap (X_2 \cup Y) = \emptyset$. Now since $\widehat{\mathcal{S}}_2 \subseteq_{rep}^{k-p_2} \mathcal{S}_2$, we have that there exists a $\widehat{X}_2 \in \widehat{\mathcal{S}}_2$ such that $\widehat{X}_1 \cup \widehat{X}_2 \cup Y \in \mathcal{I}$ and $\widehat{X}_2 \cap (\widehat{X}_1 \cup Y) = \emptyset$. This shows that $\widehat{X}_1 \cup \widehat{X}_2 \in \widehat{\mathcal{S}}_1 \bullet \widehat{\mathcal{S}}_2$ and $\widehat{X}_1 \cup \widehat{X}_2 \cup Y \in \mathcal{I}$ and thus $\widehat{\mathcal{S}}_1 \bullet \widehat{\mathcal{S}}_2 \subseteq_{rep}^{k-p_1-p_2} \mathcal{S}_1 \bullet \mathcal{S}_2$. □

The main result of this section is that given a representable matroid $M = (E, \mathcal{I})$ of rank $k = p + q$ with its representation matrix A_M, a p-family of independent sets $\mathcal{S}$, and a nonnegative weight function $w : \mathcal{S} \to \mathbb{N}$, we can compute $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$ and $\widehat{\mathcal{S}} \subseteq_{maxrep}^{q} \mathcal{S}$ of size $\binom{p+q}{p}$ deterministically in time $\mathcal{O}(\binom{p+q}{p} tp^{\omega} + t\binom{p+q}{q}^{\omega-1})$. The proof for this result is obtained by making the known exterior algebra based proof of Lovász [1977, Theorem 4.8] algorithmic. Although our proof is based on exterior algebra and is essentially the same as the proof given in Lovász [1977], we give a proof here that avoids the terminology from exterior algebra.

For our proof, we also need the following well-known generalized Laplace expansion of determinants. For a matrix $A = (a_{ij})$, the row set and the column set are denoted by $\mathbf{R}(A)$ and $\mathbf{C}(A)$, respectively. For $I \subseteq \mathbf{R}(A)$ and $J \subseteq \mathbf{C}(A)$, $A[I, J] = (a_{ij} \mid i \in I,\ j \in J)$ means the submatrix (or minor) of A with the row set I and the column set J. For $I \subseteq [n]$, let $\bar{I} = [n] \setminus I$ and $\sum I = \sum_{i \in I} i$.

PROPOSITION 3.6 (GENERALIZED LAPLACE EXPANSION). *For an $n \times n$ matrix A and $J \subseteq \mathbf{C}(A) = [n]$, it holds that*

$$\det(A) = \sum_{I \subseteq [n], |I| = |J|} (-1)^{\sum I + \sum J} \det(A[I, J]) \det(A[\bar{I}, \bar{J}]).$$

We refer to Murota [2000, Proposition 2.1.3] for a proof of the previous identity. We always assume that the number of rows in the representation matrix A_M of M over a field $\mathbb{F}$ is equal to $\mathsf{rank}(M)$=$\mathsf{rank}(A_M)$. Otherwise, using Gaussian elimination, we can obtain a matrix of the desired kind in polynomial time. See Marx [2009, Proposition 3.1] for details. We will not give the proof of Theorem 1.1, but we give a proof of the following generalization of Theorem 1.1.

THEOREM 3.7. *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 nonnegative weight function. Then there exists $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq_{maxrep}^{q} \mathcal{S}$) of size $\binom{p+q}{p}$. Moreover, given a representation A_M of M over a field $\mathbb{F}$, we can find $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq_{maxrep}^{q} \mathcal{S}$) of size at most $\binom{p+q}{p}$ in $\mathcal{O}(\binom{p+q}{p} tp^{\omega} + t\binom{p+q}{q}^{\omega-1})$ operations over $\mathbb{F}$.*

PROOF. We only show how to find $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$ in the claimed running time. The proof for $\widehat{\mathcal{S}} \subseteq_{maxrep}^{q} \mathcal{S}$ is analogous, and for that case we only point out the places where the proof differs. If $t \leq \binom{k}{p}$, then we can take $\widehat{\mathcal{S}} = \mathcal{S}$. Clearly, in this case, $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$. So from now onward we always assume that $t > \binom{k}{p}$. For the proof, we view the representation matrix A_M as a vector space over $\mathbb{F}$ and each set $S_i \in \mathcal{S}$ as a subspace of this vector space. For every element $e \in E$, let x_e be the corresponding k-dimensional column in A_M. Observe that each $x_e \in \mathbb{F}^k$. For each subspace $S_i \in \mathcal{S}$, $i \in \{1, \ldots, t\}$, we associate a

vector $\vec{s}_i = \bigwedge_{j \in S_i} x_j$ in $\mathbb{F}^{\binom{k}{p}}$ as follows. In exterior algebra terminology, the vector $\vec{s}_i$ is a wedge product of the vectors corresponding to elements in S_i. For a set $S \in \mathcal{S}$ and $I \in \binom{[k]}{p}$, we define $s[I] = \det(A_M[I, S])$.

We also define

$$\vec{s}_i = (s_i[I])_{I \in \binom{[k]}{p}}.$$

Thus, the entries of the vector $\vec{s}_i$ are the values of $\det(A_M[I, S_i])$, where I runs through all the p sized subsets of rows of A_M.

Let $H_{\mathcal{S}} = (\vec{s}_1, \ldots, \vec{s}_t)$ be the $\binom{k}{p} \times t$ matrix obtained by taking $\vec{s}_i$ as columns. Now we define a weight function $w' : \mathbf{C}(H_{\mathcal{S}}) \to \mathbb{R}^+$ on the set of columns of $H_{\mathcal{S}}$. For the column $\vec{s}_i$ corresponding to $S_i \in \mathcal{S}$, we define $w'(\vec{s}_i) = w(S_i)$. Let $\mathcal{W}$ be a set of columns of $H_{\mathcal{S}}$ that are linearly independent over $\mathbb{F}$. The size of $\mathcal{W}$ is equal to the $\mathsf{rank}(H_{\mathcal{S}})$ and is of minimum total weight with respect to the weight function w'. That is, $\mathcal{W}$ is a minimum weight column basis of $H_{\mathcal{S}}$. Since the row rank of a matrix is equal to the column rank, we have that $|\mathcal{W}| = \mathsf{rank}(H_{\mathcal{S}}) \leq \binom{k}{p}$. We define $\widehat{\mathcal{S}} = \{S_\alpha \mid \vec{s}_\alpha \in \mathcal{W}\}$. Let $|\widehat{\mathcal{S}}| = \ell$. Because $|\mathcal{W}| = |\widehat{\mathcal{S}}|$, we have that $\ell \leq \binom{k}{p}$. Without loss of generality, let $\widehat{\mathcal{S}} = \{S_i \mid 1 \leq i \leq \ell\}$ (or else we can rename these sets) and $\mathcal{W} = \{\vec{s}_1 \ldots, \vec{s}_\ell\}$. The only thing that remains to show is that indeed $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$.

Let $S_\beta \in \mathcal{S}$ be such that $S_\beta \notin \widehat{\mathcal{S}}$. We show that if there is a set $Y \subseteq E$ of size at most q such that $S_\beta \cap Y = \emptyset$ and $S_\beta \cup Y \in \mathcal{I}$, then there exists a set $\widehat{S}_\beta \in \widehat{\mathcal{S}}$ disjoint from Y with $\widehat{S}_\beta \cup Y \in \mathcal{I}$ and $w(\widehat{S}_\beta) \leq w(S_\beta)$. Let us first consider the case $|Y| = q$. Since $S_\beta \cap Y = \emptyset$, it follows that $|S_\beta \cup Y| = p + q = k$. Furthermore, since $S_\beta \cup Y \in \mathcal{I}$, we have that the columns corresponding to $S_\beta \cup Y$ in A_M are linearly independent over $\mathbb{F}$; that is, $\det(A_M[\mathbf{R}(A_M), S_\beta \cup Y]) \neq 0$.

Recall that $\vec{s}_\beta = (s_\beta[I])_{I \in \binom{[k]}{p}}$, where $s_\beta[I] = \det(A_M[I, S_\beta])$. Similarly, we define $y[L] = \det(A_M[L, Y])$ and

$$\vec{y} = (y[L])_{L \in \binom{[k]}{q}}.$$

Let $\sum J = \sum_{j \in S_\beta} j$. Define

$$\gamma(\vec{s}_\beta, \vec{y}) = \sum_{I \in \binom{[k]}{p}} (-1)^{\sum I + \sum J} s_\beta[I] \cdot y[\bar{I}].$$

Since $\binom{k}{p} = \binom{k}{k-p} = \binom{k}{q}$, the above formula is well defined. Observe that by Proposition 3.6, we have that $\gamma(\vec{s}_\beta, \vec{y}) = \det(A_M[\mathbf{R}(A_M), S_\beta \cup Y]) \neq 0$. We also know that $\vec{s}_\beta$ can be written as a linear combination of vectors in $\mathcal{W} = \{\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_\ell\}$. That is, $\vec{s}_\beta = \sum_{i=1}^{\ell} \lambda_i \vec{s}_i$, $\lambda_i \in \mathbb{F}$, and for some i, $\lambda_i \neq 0$. Thus,

$$\begin{aligned}
\gamma(\vec{s}_\beta, \vec{y}) &= \sum_I (-1)^{\sum I + \sum J} s_\beta[I] \cdot y[\bar{I}] \\
&= \sum_I (-1)^{\sum I + \sum J} \left(\sum_{i=1}^{\ell} \lambda_i s_i[I] \right) y[\bar{I}]
\end{aligned}$$

$$= \sum_{i=1}^{\ell} \lambda_i \left(\sum_{I} (-1)^{\sum I + \sum J} s_i[I] y[\bar{I}] \right)$$
$$= \sum_{i=1}^{\ell} \lambda_i \det(A_M[\mathbf{R}(A_M), S_i \cup Y]) \quad \text{(by Proposition 3.6).}$$

Define

$$\mathsf{sup}(S_\beta) = \left\{ S_i \;\middle|\; S_i \in \widehat{\mathcal{S}},\ \lambda_i \det(A_M[\mathbf{R}(A_M), S_i \cup Y])) \neq 0 \right\}.$$

Since $\gamma(\vec{s}_\beta, \vec{y}) \neq 0$, we have that $(\sum_{i=1}^{\ell} \lambda_i \det(A_M[\mathbf{R}(A_M), S_i \cup Y])) \neq 0$ and thus $\mathsf{sup}(S_\beta) \neq \emptyset$. Observe that for all $S \in \mathsf{sup}(S_\beta)$, we have that $\det(A_M[\mathbf{R}(A_M), S \cup Y]) \neq 0$ and thus $S \cup Y \in \mathcal{I}$. We now show that $w(S) \leq w(S_\beta)$ for all $S \in \mathsf{sup}(S_\beta)$.

Claim 3.1. *For all $S \in \mathsf{sup}(S_\beta)$, $w(S) \leq w(S_\beta)$.*

Proof. For a contradiction, assume that there exists a set $S_j \in \mathsf{sup}(S_\beta)$ such that $w(S_j) > w(S_\beta)$. Let $\vec{s}_j$ be the vector corresponding to S_j and $\mathcal{W}' = (\mathcal{W} \cup \{\vec{s}_j\}) \setminus \{\vec{s}_\beta\}$. Since $w(S_j) > w(S_\beta)$, we have that $w(\vec{s}_j) > w(\vec{s}_\beta)$ and thus $w'(\mathcal{W}) > w'(\mathcal{W}')$. Now we show that $\mathcal{W}'$ is also a column basis of H_S. This will contradict our assumption that $\mathcal{W}$ is a minimum weight column basis of H_S. Recall that $\vec{s}_\beta = \sum_{i=1}^{\ell} \lambda_i \vec{s}_i$, $\lambda_i \in \mathbb{F}$. Since $S_j \in \mathsf{sup}(S_\beta)$, we have that $\lambda_j \neq 0$. Thus, $\vec{s}_j$ can be written as a linear combination of vectors in $\mathcal{W}'$. That is,

$$\vec{s}_j = \lambda_\beta \vec{s}_\beta + \sum_{i=1, i \neq j}^{\ell} \lambda'_i \vec{s}_i. \tag{1}$$

Also, every vector $\vec{s}_\gamma \notin \mathcal{W}$ can be written as a linear combination of vectors in $\mathcal{W}$:

$$\vec{s}_\gamma = \sum_{i=1}^{\ell} \delta_i \vec{s}_i, \quad \delta_i \in \mathbb{F}. \tag{2}$$

By substituting Equation (1) into Equation (2), we conclude that every vector can be written as a linear combination of vectors in $\mathcal{W}'$. This shows that $\mathcal{W}'$ is also a column basis of H_S, a contradiction proving the claim. □

Claim 3.1 and the discussions preceding it show that we could take any set $S \in \mathsf{sup}(S_\beta)$ as the desired $\widehat{S}_\beta \in \widehat{\mathcal{S}}$. Also, since $\det(A_M[\mathbf{R}(A_M), S \cup Y]) \neq 0$, we have that $S \cap Y = \emptyset$. This shows that indeed $\widehat{\mathcal{S}} \subseteq^{q}_{minrep} \mathcal{S}$ for each Y of size q. This completes the proof for the case $|Y| = q$.

Suppose that $|Y| = q' < q$. Since M is a matroid of rank $k = p + q$, there exists a superset $Y' \in \mathcal{I}$ of Y of size q such that $S_\beta \cap Y' = \emptyset$ and $S_\beta \cup Y' \in \mathcal{I}$. This implies that there exists a set $\widehat{S} \in \widehat{\mathcal{S}}$ such that $\det(A_M[\mathbf{R}(A_M), \widehat{S} \cup Y']) \neq 0$ and $w(\widehat{S}) \leq w(S)$. Thus, the columns corresponding to $\widehat{S} \cup Y$ are linearly independent.

We now consider the running time of the algorithm. To make the previous proof algorithmic, we need to

(a) compute determinants and
(b) apply fast Gaussian elimination to find a minimum weight column basis.

It is well known that one can compute the determinant of an $n \times n$ matrix in time $\mathcal{O}(n^\omega)$ [Bunch and Hopcroft 1974]. For a rectangular matrix A of size $d \times n$ (with $d \leq n$), Bodlaender et al. [2015] outline an algorithm computing a minimum weight column

basis in time $\mathcal{O}(nd^{\omega-1})$. Thus, given a p-family of independent sets $\mathcal{S}$, we can construct the matrix $H_{\mathcal{S}}$ as follows. For every set S_i, we first compute $\vec{s}_i$. To do this, we compute $\det(A_M[I, S_i])$ for every $I \in \binom{[k]}{p}$. This can be done in time $\mathcal{O}(\binom{p+q}{p}p^{\omega})$. Thus, we can obtain the matrix $H_{\mathcal{S}}$ in time $\mathcal{O}(\binom{p+q}{p}tp^{\omega})$. Given matrix $H_{\mathcal{S}}$, we can find a minimum weight column basis $\mathcal{W}$ of $H_{\mathcal{S}}$ in time $\mathcal{O}(t\binom{p+q}{p}^{\omega-1})$. Given $\mathcal{W}$, we can easily recover $\widehat{\mathcal{S}}$. Thus, we can compute $\widehat{\mathcal{S}} \subseteq^{q}_{minrep} \mathcal{S}$ in $\mathcal{O}(\binom{p+q}{p}tp^{\omega} + t\binom{p+q}{q}^{\omega-1})$ field operations. This concludes the proof for finding $\widehat{\mathcal{S}} \subseteq^{q}_{minrep} \mathcal{S}$. To find $\widehat{\mathcal{S}} \subseteq^{q}_{maxrep} \mathcal{S}$, the only change we need to do in the algorithm for finding $\widehat{\mathcal{S}} \subseteq^{q}_{minrep} \mathcal{S}$ is to find a *maximum weight column basis* $\mathcal{W}$ of $H_{\mathcal{S}}$. This concludes the proof. □

In Theorem 3.7, we assumed that $\mathsf{rank}(M) = p+q$. However, one can obtain a similar result even when $\mathsf{rank}(M) > p+q$ in lieu of randomness. To do this, we first need to compute the representation matrix of a k-restriction of $M = (E, \mathcal{I})$. For that, we make use of Proposition 2.6. This step returns a representation of a k-restriction of $M = (E, \mathcal{I})$ with high probability. Given this matrix, we apply Theorem 3.7 and arrive at the following result.

THEOREM 3.8. *Let $M = (E, \mathcal{I})$ be a linear matroid, $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a p-family of independent sets, and $w : \mathcal{S} \to \mathbb{N}$ be a nonnegative weight function. Then there exists $\widehat{\mathcal{S}} \subseteq^{q}_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^{q}_{maxrep} \mathcal{S}$) of size $\binom{p+q}{p}$. Furthermore, given a representation A_M of M over a field $\mathbb{F}$, there is a randomized algorithm computing $\widehat{\mathcal{S}} \subseteq^{q}_{minrep} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq^{q}_{maxrep} \mathcal{S}$) of size at most $\binom{p+q}{p}$ in $\mathcal{O}(\binom{p+q}{p}tp^{\omega} + t\binom{p+q}{q}^{\omega-1} + ||A_M||^{\mathcal{O}(1)})$ operations over $\mathbb{F}$, where $||A_M||$ denotes the length of A_M in the input.*

4. FAST COMPUTATION FOR REPRESENTATIVE FAMILIES FOR UNIFORM MATROIDS

In this section, we show that for uniform matroids, one can avoid matrix multiplication computations in order to compute representative families. The section is organized as follows. We start (Section 4.1, Theorem 4.1) from a relatively simple algorithm computing representative families over a uniform matroid. This algorithm is already faster than the algorithm of Theorem 1.1 for general matroids. In Section 4.2, Theorem 4.15, we give an even faster but more complicated algorithm. Throughout this section, a subfamily $\mathcal{A}' \subseteq \mathcal{A}$ of the family $\mathcal{A}$ is said to *q-represent* $\mathcal{A}$ if for every set B of size q such that there is an $A \in \mathcal{A}$ and $A \cap B = \emptyset$, there is a set $A' \in \mathcal{A}'$ such that $A' \cap B = \emptyset$.

4.1. Representative Families Using Lopsided Universal Sets

Our aim in this subsection is to prove the following theorem.

THEOREM 4.1. *There is an algorithm that, given a family $\mathcal{A}$ of p-sets over a universe U of size n and an integer q, computes in time $|\mathcal{A}| \cdot \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ a subfamily $\mathcal{A}' \subseteq A$ such that $|\mathcal{A}'| \leq \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ and $\mathcal{A}'$ q-represents $\mathcal{A}$.*

The main tool in our proof of Theorem 4.1 is a generalization of the notion of *n-k-universal* families. A family $\mathcal{F}$ of sets over a universe U is an *n-k-universal* family if for every set $A \in \binom{U}{k}$ and every subset $A' \subseteq A$, there is some set $F \in \mathcal{F}$ whose intersection $F \cap A$ is exactly A'. Naor et al. [1995] show that given n and k, one can construct an n-k-universal family $\mathcal{F}$ of size $2^{k+o(k)} \cdot \log n$ in time $2^{k+o(k)} \cdot n \log n$.

We tweak the notion of universal families as follows. We will say that a family $\mathcal{F}$ of sets over a universe U of size n is an *n-p-q-lopsided-universal* family if for every $A \in \binom{U}{p}$ and $B \in \binom{U \setminus A}{q}$, there is an $F \in \mathcal{F}$ such that $A \subseteq F$ and $B \cap F = \emptyset$. An

alternative definition that is easily seen to be equivalent is that $\mathcal{F}$ is n-p-q-lopsided-universal if for every subset $A \in \binom{U}{p+q}$ and every subset $A' \in \binom{A}{p}$, there is an $F \in \mathcal{F}$ such that $F \cap A = A'$. From the second definition, it follows that an n-$(p+q)$-universal family is also n-p-q-lopsided-universal. Thus, the construction of Naor et al. [1995] of universal set families also gives a construction of an n-p-q-lopsided universal family of size $2^{p+q+o(p+q)} \cdot \log n$, running in time $2^{p+q+o(p+q)} \cdot n \log n$. It turns out that by slightly changing the construction of Naor et al. [1995], one can prove the following result.

LEMMA 4.2. *There is an algorithm that given n, p and q, constructs an n-p-q-lopsided-universal family $\mathcal{F}$ of size $\binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ in time $\mathcal{O}(\binom{p+q}{p} \cdot 2^{o(p+q)} \cdot n \log n)$.*

We do not give a stand-alone proof of Lemma 4.2; however, Lemma 4.2 is a direct corollary of Lemma 4.4 proved in Section 4.2. We will now show how to use the lemma to prove Theorem 4.1.

PROOF OF THEOREM 4.1. The algorithm starts by constructing an n-p-q-lopsided universal family $\mathcal{F}$ as guaranteed by Lemma 4.2. If $|\mathcal{A}| \leq |\mathcal{F}|$, the algorithm outputs $\mathcal{A}$ and halts. Otherwise, it builds the set $\mathcal{A}'$ as follows. Initially $\mathcal{A}'$ is equal to $\emptyset$ and all sets in $\mathcal{F}$ are marked as unused. The algorithm goes through every $A \in \mathcal{A}$ and unused sets $F \in \mathcal{F}$. If an unused set $F \in \mathcal{F}$ is found such that $A \subseteq F$, the algorithm marks F as used, inserts A into $\mathcal{A}'$, and proceeds to the next set in $\mathcal{A}$. If no such set F is found, the algorithm proceeds to the next set in $\mathcal{A}$ without inserting A into $\mathcal{A}'$.

The size of $\mathcal{A}'$ is upper bounded by $|\mathcal{F}| \leq \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ since every time a set is added to $\mathcal{A}'$, an unused set in $\mathcal{F}$ is marked as used. For the running time analysis, constructing $\mathcal{F}$ takes time $\binom{p+q}{p} \cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot n \log n$. Then we run through all of $\mathcal{F}$ for each set $A \in \mathcal{A}$, spending time $|\mathcal{A}| \cdot |\mathcal{F}| \cdot (p+q)^{\mathcal{O}(1)}$, which is at most $|\mathcal{A}| \cdot \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$. Thus, in total, the running time is bounded by $|\mathcal{A}| \cdot \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$.

Finally, we need to argue that $\mathcal{A}'$ q-represents $\mathcal{A}$. Consider any set $A \in \mathcal{A}$ and B such that $|B| = q$ and $A \cap B = \emptyset$. If $A \in \mathcal{A}'$, we are done, so assume that $A \notin \mathcal{A}'$. Since $\mathcal{F}$ is n-p-q-lopsided universal, there is a set $F \in \mathcal{F}$ such that $A \subseteq F$ and $F \cap B = \emptyset$. Since $A \notin \mathcal{A}'$, we know that F has already been marked as used when A was considered by the algorithm. When the algorithm marked F as used, it also inserted a set A' into $\mathcal{A}'$. For the insertion to be made, F must satisfy $A' \subseteq F$. But then $A' \cap B = \emptyset$, completing the proof. □

One of the factors that drives up the running time of the algorithm in Theorem 4.1 is that one needs to consider all of $\mathcal{F}$ for each set $A \in \mathcal{A}$. Doing some computations, it is possible to convince oneself that in an n-p-q-lopsided universal family $\mathcal{F}$, the number of sets $F \in \mathcal{F}$ containing a fixed set A of size p should be approximately $|\mathcal{F}| \cdot (\frac{p}{p+q})^p$. Thus, if we could only make sure that this estimation is in fact correct for every $A \in \mathcal{A}$ *and* we could make sure that for a given $A \in \mathcal{A}$ we can list all of the sets in $\mathcal{F}$ that contain A without having to go through the sets that do not, then we could speed up our algorithm by a factor $(\frac{p+q}{p})^p$. This is exactly the strategy behind the main theorem of Section 4.2.

4.2. Representative Families Using Separating Collections

In this section, we design a faster algorithm to find a q-representative family. Our main technical tool is a construction of an *n-p-q-separating collection*. We start with the formal definition of an *n-p-q-separating collection*.

Definition 4.3. An n-p-q-separating collection $\mathcal{C}$ is a tuple $(\mathcal{F}, \chi, \chi')$, where $\mathcal{F}$ is a family of sets over a universe U of size n, χ is a function from $\bigcup_{p' \leq p} \binom{U}{p'}$ to $2^{\mathcal{F}}$, and χ' is a function from $\bigcup_{q' \leq q} \binom{U}{q'}$ to $2^{\mathcal{F}}$ such that the following properties are satisfied:

(1) For every $A \in \bigcup_{p' \leq p} \binom{U}{p'}$ and $F \in \chi(A)$, $A \subseteq F$.
(2) For every $B \in \bigcup_{q' \leq q} \binom{U}{q'}$ 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 $p_1 + \cdots + p_r = p$, $\exists F \in \chi(A_1) \cap \chi(A_2) \cdots \chi(A_r) \cap \chi'(B)$.

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

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

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

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

We must remark that the definition of an n-p-q-separating collection in the preliminary version of this article [Fomin et al. 2014a] was slightly more restricted than the one given here. This new definition has already been used to obtain faster algorithms for computing representative families for product families [Fomin et al. 2014b].

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 $A \in \bigcup_{p' \leq p} \binom{U}{p'}$ or $B \in \bigcup_{q' \leq q} \binom{U}{q'}$, 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 $\mathcal{C} = (\mathcal{F}, \chi, \chi')$ computed by the data structure should form an *n-p-q-separating collection*.

We call the time the data structure takes to initialize and output $\mathcal{F}$ the *initialization time*. The *(χ, p')-query time*, $p' \leq p$, of the data structure is the maximum time the data structure uses to compute $\chi(A)$ over all $A \in \binom{U}{p'}$. Similarly, the *(χ', q')-query time*, $q' \leq q$, of the data structure is the maximum time the data structure uses to compute $\chi'(B)$ over all $B \in \binom{U}{q'}$. The initialization time of the data structure and the size of $\mathcal{C}$ are functions of n, p, and q. The initialization time is denoted by $\tau_I(n, p, q)$, and the size of $\mathcal{C}$ is denoted by $\zeta(n, p, q)$. The (χ, p')-query time and (χ, p')-degree of $\mathcal{C}$, $p' \leq p$, are functions of n, p', p, q and are denoted by $Q_{(\chi,p')}(n, p, q)$ and $\Delta_{(\chi,p')}(n, p, q)$, respectively. Similarly, the (χ', q')-query time and (χ', q')-degree of $\mathcal{C}$, $q' \leq q$, are functions of n, q', p, q and are denoted by $Q_{(\chi',q')}(n, p, q)$ and $\Delta_{(\chi',q')}(n, p, q)$, respectively. We are now ready to state the main technical tool of this subsection.

Lemma 4.4. *Given $0 < x < 1$, there is a construction of n-p-q-separating collection with the following parameters:*

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

—(χ', q')-*degree,* $\Delta_{(\chi',q')}(n, p, q) \leq 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot \frac{1}{x^p(1-x)^{q-q'}} \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n$
—(χ', q')-*query time,* $Q_{(\chi',q')}(n, p, q) \leq 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot \frac{1}{x^p(1-x)^{q-q'}} \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n$

We first give the road map that we take to prove Lemma 4.4. The proof of Lemma 4.4 uses three auxiliary lemmata.

(a) **Existential Proof (Lemma 4.5).** This lemma shows that there is indeed an n-p-q-separating collection with the required sizes, degrees, and query time. Essentially, it shows that if we form a family $\mathcal{F} = \{F_1, \ldots, F_t\}$ of sets of U such that each F_i is a random subset of U where each element is inserted into F_i with probability x, then $\mathcal{F}$ has the desired sizes, degrees, and query time. Thus, this also gives a brute-force algorithm to design the family $\mathcal{F}$ by just guessing the family of desired size and then checking whether it is indeed an n-p-q-separating collection.
(b) **Universe Reduction (Lemma 4.8).** The construction obtained in Lemma 4.5 has only one drawback that the initialization time is much larger than claimed in Lemma 4.4. To overcome this lacuna, we do not apply the construction in Lemma 4.5 directly. We first prove Lemma 4.8, which helps us in reducing the universe size to $(p+q)^2$. This is done using the known construction of k-perfect hash families of size $(p+q)^{\mathcal{O}(1)} \log n$. However, Lemma 4.8 alone cannot reduce the universe size sufficiently that we can apply the construction of Lemma 4.5.
(c) **Splitting Lemma (Lemma 4.11).** We give a splitter-type construction in Lemma 4.11 that when applied with Lemma 4.8 makes the universe and other parameters small enough that we can apply the construction given in Lemma 4.5. In this construction, we consider all the "consecutive partitions" of the universe into t parts; assume that the sets $A \cup B$, $A = \cup_{i=1}^{r} A_i$ are distributed uniformly into t parts; use this information to obtain a construction of separating collections in each part; and then take the product of these collections to obtain a collection for the original instance.

We start with the existential proof.

LEMMA 4.5. *Given* $0 < x < 1$*, there is a construction of* n-p-q-*separating collections with*

—*size* $\zeta(n, p, q) = \mathcal{O}(\frac{1}{x^p(1-x)^q} \cdot (p^2+q^2+1)\log n)$*;*
—*initialization time* $\tau_I(n, p, q) = \mathcal{O}(\binom{2^n}{\zeta(n,p,q)} \cdot \frac{1}{x^p(1-x)^q} \cdot n^{\mathcal{O}(p+q)})$*;*
—(χ, p')-*degree for* $p' \leq p$, $\Delta_{(\chi,p')}(n, p, q) = \mathcal{O}(\frac{1}{x^{p-p'}} \cdot \frac{(p^2+q^2+1)}{(1-x)^q} \cdot \log n)$*;*
—(χ, p')-*query time* $Q_{(\chi,p')}(n, p, q) = \mathcal{O}(\frac{1}{x^p(1-x)^q} \cdot n^{\mathcal{O}(1)})$*;*
—(χ', q')-*degree* $\Delta_{(\chi',q')}(n, p, q) = \mathcal{O}(\frac{1}{x^p(1-x)^{q-q'}} \cdot (p^2+q^2+1) \cdot \log n)$*; and*
—(χ', q')-*query time* $Q_{(\chi',q')}(n, p, q) = \mathcal{O}(\frac{1}{x^p(1-x)^q} \cdot n^{\mathcal{O}(1)})$.

PROOF. We start by giving a randomized algorithm that with positive probability constructs an n-p-q-separating collection $\mathcal{C} = (\mathcal{F}, \chi, \chi')$ with the desired size and degree parameters. We will then discuss how to deterministically compute such a $\mathcal{C}$ within the required time bound. Set $t = \frac{1}{x^p(1-x)^q} \cdot (p^2+q^2+1)\log n$ and construct the family $\mathcal{F} = \{F_1, \ldots, F_t\}$ as follows. Each set F_i is a random subset of U, where each element of U is inserted into F_i with probability x. Distinct elements are inserted (or not) into F_i independently, and the construction of the different sets in $\mathcal{F}$ is also independent. For each $A \in \bigcup_{p' \leq p} \binom{U}{p'}$, we set $\chi(A) = \{F \in \mathcal{F} : A \subseteq F\}$, and for each $B \in \bigcup_{q' \leq q} \binom{U}{q'}$, we set $\chi'(B) = \{F \in \mathcal{F} : F \cap B = \emptyset\}$.

The size of $\mathcal{F}$ is within the required bound by construction. We now argue that with positive probability, $(\mathcal{F}, \chi, \chi')$ is indeed an n-p-q-separating collection, and that the degrees of $\mathcal{C}$ are within the required bounds as well. For fixed sets $A \in \binom{U}{p}$ and $B \in \binom{U \setminus A}{q}$, and for integer $i \leq t$, we consider the probability that $A \subseteq F_i$ and $B \cap F_i = \emptyset$. This probability is $x^p(1-x)^q$. Since each F_i is constructed independently from the other sets in $\mathcal{F}$, the probability that *no* F_i satisfies $A \subseteq F_i$ and $B \cap F_i = \emptyset$ is

$$\left(1 - x^p(1-x)^q\right)^t \leq e^{-(p^2+q^2+1)\log n} = \frac{1}{n^{p^2+q^2+1}}.$$

For a fixed $A_1, \ldots, A_r$ and B (choices in condition 3), the probability that no F_i is in $\chi(A_1) \cap \chi(A_2) \cap \cdots \cap \chi(A_r) \cap \chi'(B)$ is equal to the probability that no F_i is in $\chi(A_1 \cup A_2 \cdots \cup A_r) \cap \chi'(B)$ (since $\chi(A')$ contains all the sets in $\mathcal{F}$ that contain A' and $\chi'(B)$ contains all the sets in $\mathcal{F}$ that are disjoint from B). Hence, the probability that condition 3 fails is upper bounded by

$$Y \cdot \frac{1}{n^{p^2+q^2+1}},$$

where Y is the number of choices for $A_1, \ldots, A_r$ and B in condition 3. We upper bound Y as follows. There are $\binom{n}{p}$ choices for $A_1 \cup \cdots \cup A_r$ and $\binom{n}{q}$ choices for B. For each choice of $A_1 \cup \cdots \cup A_r$, there are at most r^p choices of making $A_1, \ldots, A_r$, with some of them being empty as well. Note that $r \leq p$. Therefore, the number of possible choices of sets $A_1, A_2, \ldots, A_r$ and B in condition 3 is upper bounded by $\binom{n}{p}\binom{n}{q}p^p \leq n^{2p+q} \leq n^{p^2+q^2}$. Hence, the probability that condition 3 in Definition 4.3 fails is at most $\frac{1}{n}$.

We also need to upper bound the maximum degree of $\mathcal{C}$. For every $A \in \binom{U}{p'}$, $|\chi(A)|$ is a random variable. For a fixed $A \in \binom{U}{p'}$ and $i \leq t$, the probability that $A \subseteq F_i$ is exactly $x^{p'}$. Hence, $|\chi(A)|$ is the sum of t independent 0/1-random variables that each take value 1 with probability $x^{p'}$. Hence, the expected value of $|\chi(A)|$ is

$$E[|\chi(A)|] = t \cdot x^{p'} = \frac{1}{x^{p-p'}(1-x)^q} \cdot (p^2+q^2+1)\log n.$$

For every $B \in \binom{U}{q'}$, $|\chi'(B)|$ is also a random variable. For a fixed $B \in \binom{U}{q'}$ and $i \leq t$, the probability that $A \cap F_i = \emptyset$ is exactly $(1-x)^{q'}$. Hence, the expected value of $|\chi'(B)|$ is

$$E[|\chi'(B)|] = t \cdot (1-x)^{q'} = \frac{1}{x^p(1-x)^{q-q'}} \cdot (p^2+q^2+1)\log n.$$

Standard Chernoff bounds [Mitzenmacher and Upfal 2005, Theorem 4.4] show that the probability that for any $A \in \binom{U}{p'}$, $|\chi(A)|$ is at least $6E[|\chi(A)|]$ is upper bounded by $2^{-6E[|\chi(A)|]} \leq \frac{1}{n^{p^2+q^2+1}}$. Similarly, the probability that for any $B \in \binom{U}{q'}$, $|\chi'(B)|$ is at least $6E[|\chi'(B)|]$ is upper bounded by $2^{-6E[|\chi'(B)|]} \leq \frac{1}{n^{p^2+q^2+1}}$. There are $\sum_{p' \leq p}\binom{n}{p'} \leq n^{p^2}$ choices for $A \in \bigcup_{p' \leq p}\binom{U}{p'}$ and $\sum_{q' \leq q}\binom{n}{q'} \leq n^{q^2}$ choices for $B \in \bigcup_{q' \leq q}\binom{U}{q'}$. Hence, the union bound yields that the probability that there exists an $A \in \bigcup_{p' \leq p}\binom{U}{p'}$ such that $|\chi(A)| > 6E[|\chi(A)|]$ or there exists $B \in \bigcup_{q' \leq q}\binom{U}{q'}$ such that $|\chi'(B)| > 6E[|\chi'(B)|]$ is upper bounded by $\frac{1}{n}$. Thus, $\mathcal{C}$ is a family of n-p-q-separating collections with the desired size and degree parameters with probability at least $1 - \frac{2}{n} > 0$. The degenerate case that $1 - \frac{2}{n} \leq 0$ is handled by the family $\mathcal{F}$ containing all (at most four) subsets of U.

To construct $\mathcal{F}$ within the stated initialization time bound, it is sufficient to try all families $\mathcal{F}$ of size t and, for each of the $\binom{2^n}{\zeta(n,p,q)}$ guesses, test whether it is indeed a family of n-p-q-separating collections in time $\mathcal{O}(t \cdot n^{\mathcal{O}(p+q)}) = \mathcal{O}(\frac{1}{x^p(1-x)^q} \cdot n^{\mathcal{O}(p+q)})$.

For the queries, we need to give an algorithm that, given A, computes $\chi(A)$ (or $\chi'(A)$), under the assumption that $\mathcal{F}$ has already has been computed in the initialization step. This is easily done within the stated running time bound by going through every set $F \in \mathcal{F}$, checking whether $A \subseteq F$ (or $A \cap F = \emptyset$), and, if so, inserting F into $\chi(A)$ ($\chi'(A)$). This concludes the proof. □

We will now work toward improving the time bounds of Lemma 4.5. To that end, we will need a construction of *k-perfect hash functions* by Alon et al. [1995].

Definition 4.6. A family of functions $f_1, \ldots, f_t$ from a universe U of size n to a universe of size r is a k-perfect family of hash functions if for every set $S \subseteq U$ such that $|S| = k$, there exists an i such that the restriction of f_i to S is injective.

Alon et al. [1995] give very efficient constructions of k-perfect families of hash functions from a universe of size n to a universe of size k^2.

PROPOSITION 4.7 (ALON ET AL. [1995]). *For any universe U of size n, there is a k-perfect family $f_1, \ldots, f_t$ of hash functions from U to $[k^2]$ with $t = \mathcal{O}(k^{\mathcal{O}(1)} \cdot \log n)$. Such a family of hash functions can be constructed in time $\mathcal{O}(k^{\mathcal{O}(1)} n \log n)$.*

LEMMA 4.8. *If there is a construction of n-p-q-separating collections $(\hat{\mathcal{F}}, \hat{\chi}, \hat{\chi}')$ with initialization time $\tau_I(n,p,q)$, size $\zeta(n,p,q)$, $(\hat{\chi}, p')$-query time $Q_{(\hat{\chi},p')}(n,p,q)$, $(\hat{\chi}', q')$-query time $Q_{(\hat{\chi}',q')}(n,p,q)$, $(\hat{\chi}, p')$-degree $\Delta_{(\hat{\chi},p')}(n,p,q)$, and $(\hat{\chi}', q')$-degree $\Delta_{(\hat{\chi}',q')}(n,p,q)$, then there is a construction of n-p-q-separating collections with the following parameters:*

—$\zeta'(n,p,q) \leq \zeta((p+q)^2, p, q) \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n$,
—$\tau'_I(n,p,q) = \mathcal{O}(\tau_I((p+q)^2, p, q) + \zeta((p+q)^2, p, q) \cdot (p+q)^{\mathcal{O}(1)} \cdot n \log n)$,
—$\Delta'_{(\chi,p')}(n,p,q) \leq \Delta_{(\hat{\chi},p')}((p+q)^2, p, q) \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n$,
—$Q'_{(\chi,p')}(n,p,q) = \mathcal{O}((Q_{(\hat{\chi},p')}((p+q)^2, p, q) + \Delta_{(\hat{\chi},p')}((p+q)^2, p, q)) \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$,
—$\Delta'_{(\chi',q')}(n,p,q) \leq \Delta_{(\hat{\chi}',q')}((p+q)^2, p, q) \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n$,
—$Q'_{(\chi',q')}(n,p,q) = \mathcal{O}((Q_{(\hat{\chi}',q')}((p+q)^2, p, q) + \Delta_{(\hat{\chi}',q')}((p+q)^2, p, q)) \cdot (p+q)^{\mathcal{O}(1)} \log n)$.

PROOF. We give a construction of n-p-q-separating collections with initialization time, query time, size, and degree of τ'_I, Q', ζ', and Δ', respectively, using the construction with initialization time, query time, size, and degree of τ_I, Q, ζ, and Δ as a black box.

We first describe the initialization of the data structure. Given n, p, and q, we construct using Proposition 4.7 a $(p+q)$-perfect family $f_1, \ldots f_t$ of hash functions from the universe U to $[(p+q)^2]$. The construction takes time $\mathcal{O}((p+q)^{\mathcal{O}(1)} n \log n)$ and $t \leq (p+q)^{\mathcal{O}(1)} \cdot \log n$. We will store these hash functions in memory. We use the following notations:

—For a set $S \subseteq U$ and $T \subseteq [(p+q)^2]$,
$f_i(S) = \{f_i(s) : s \in S\}$ and $f_i^{-1}(T) = \{s \in U : f(s) \in T\}$.
—For a family $\mathcal{Z}$ of sets over U and family $\mathcal{W}$ of sets over $[(p+q)^2]$,
$f_i(\mathcal{Z}) = \{f_i(S) : S \in \mathcal{Z}\}$ and $f_i^{-1}(\mathcal{W}) = \{f_i^{-1}(T) : T \in \mathcal{W}\}$.

We first use the given black-box construction for $(p+q)^2$-p-q-separating collections $(\hat{\mathcal{F}}, \hat{\chi}, \hat{\chi}')$ over the universe $[(p+q)^2]$. We run the initialization algorithm of this

construction and store the family $\hat{\mathcal{F}}$ in memory. We then set

$$\mathcal{F} = \bigcup_{i \le t} f_i^{-1}(\hat{\mathcal{F}}).$$

We spent $\mathcal{O}((p+q)^{\mathcal{O}(1)} n \log n)$ time to construct a $(p+q)$-perfect family of hash functions, $\mathcal{O}(\tau_I((p+q)^2, p, q))$ to construct $\hat{\mathcal{F}}$ of size $\zeta((p+q)^2, p, q)$, and $\mathcal{O}(\zeta((p+q)^2, p, q) \cdot (p+q)^{\mathcal{O}(1)} \cdot n \log n)$ time to construct $\mathcal{F}$ from $\hat{\mathcal{F}}$ and the family of perfect hash functions. Thus, the upper bound on $\tau_I'(n, p, q)$ follows. Furthermore, $|\mathcal{F}| \le |\hat{\mathcal{F}}| \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n$, yielding the claimed bound for ζ'.

We now define $\chi(A)$ for every $A \in \bigcup_{p' \le p} \binom{U}{p'}$ and describe the query algorithm. For every $A \in \bigcup_{p' \le p} \binom{U}{p'}$, we let

$$\chi(A) = \bigcup_{\substack{i \le t \\ |f_i(A)| = |A|}} f_i^{-1}(\hat{\chi}(f_i(A))).$$

Since for every $\hat{F} \in \hat{\chi}(f_i(A))$, $f_i(A) \subseteq \hat{F}$, it follows that $A \subseteq F$ for every $F \in \chi(A)$. Furthermore, we can bound $|\chi(A)|$ for any $A \in \bigcup_{p' \le p} \binom{U}{p'}$, as follows:

$$|\chi(A)| \le \sum_{\substack{i \le t \\ |f_i(A)| = |A|}} |\hat{\chi}(f_i(A))| \le \Delta_{(\hat{\chi}, p')}((p+q)^2, p, q) \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n.$$

Thus, the claimed bound for $\Delta'_{(\chi, p')}$ follows. Similarly, way can define $\chi'(B)$ for every $B \in \bigcup_{q' \le q} \binom{U}{q'}$ as

$$\chi'(B) = \bigcup_{\substack{i \le t \\ |f_i(A)| = |A|}} f_i^{-1}(\hat{\chi}'(f_i(A))).$$

$$|\chi'(B)| \le \sum_{\substack{i \le t \\ |f_i(A)| = |A|}} |\hat{\chi}'(f_i(A))| \le \Delta_{(\hat{\chi}', q')}((p+q)^2, p, q) \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n.$$

To compute $\chi(A)$ for any $A \in \bigcup_{p' \le p} \binom{U}{p'}$, we go over every $i \le t$ and check whether f_i is injective on A. This takes time $\mathcal{O}((p+q)^{\mathcal{O}(1)} \cdot \log n)$. For each i such that f_i is injective on A, we compute $f_i(A)$ and then $\hat{\chi}(f_i(A))$ in time $\mathcal{O}(Q_{(\hat{\chi}, p')}((p+q)^2, p, q))$. Then we compute $f_i^{-1}(\hat{\chi}(f_i(A)))$ in time $\mathcal{O}(|\hat{\chi}(f_i(A))| \cdot (p+q)^{\mathcal{O}(1)}) = \mathcal{O}(\Delta_{(\hat{\chi}, p')}((p+q)^2, p, q) \cdot (p+q)^{\mathcal{O}(1)})$ and add this set to $\chi(A)$. As we need to do this $\mathcal{O}((p+q)^{\mathcal{O}(1)} \cdot \log n)$ times, the total time to compute $\chi(A)$ is upper bounded by $\mathcal{O}((Q_{(\hat{\chi}, p')}((p+q)^2, p, q) + \Delta_{(\hat{\chi}, p')}((p+q)^2, p, q)) \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$, yielding the claimed upper bound on $Q'_{(\chi, p')}$. In a similar way we can bound $Q'_{(\chi', q')}$.

It remains to argue that $(\mathcal{F}, \chi, \chi')$ is in fact an n-p-q-separating collection. For any r, consider pairwise disjoint sets $A_1 \in \binom{U}{p_1}, \ldots, A_r \in \binom{U}{p_r}$ and $B \in \binom{U}{q}$ such that $p_1 + \cdots + p_r = p$. We need to show that there is $F \in \chi(A_1) \cap \cdots \cap \chi(A_r) \cap \chi'(B)$. Since $f_1, \ldots, f_t$ is a $(p+q)$-perfect family of hash functions, there is an i such that f_i is injective on $A_1 \cup \cdots \cup A_r \cup B$. Since $(\hat{\mathcal{F}}, \hat{\chi}, \hat{\chi}')$ is a $(p+q)^2$-p-q-separating collection, $\exists \hat{F} \in \hat{\chi}(f_i(A_1)) \cap \cdots \hat{\chi}(f_i(A_r)) \cap \hat{\chi}'(f_i(B))$. Since f_i is injective on $A_1, \ldots, A_r$ and B, $f_i^{-1}(\hat{F}) \in \chi(A_1) \cap \cdots \chi(A_r) \cap \chi'(B)$. This concludes the proof. □

We now give a *splitting lemma*, which allows us to reduce the problem of finding n-p-q-separating collections to the same problem, but with much smaller values for p and q.

A *partition* of U is a family $\mathcal{U}_P = \{U_1, U_2, \ldots U_t\}$ of sets over U such that $U_i \cap U_j = \emptyset$ for every $i \neq j$ and $U = \bigcup_{i \leq t} U_i$. Each of the sets U_i are called the *parts* of the partition. A *consecutive partition* of $\{1, \ldots, n\}$ is a partition $\mathcal{U}_P = \{U_1, U_2, \ldots U_t\}$ of $\{1, \ldots, n\}$ such that for every integer $i \leq t$ and integers $1 \leq x \leq y \leq z$, if $x \in U_i$ and $z \in U_i$, then $y \in U_i$ as well. In other words, in a consecutive partition, each part is a consecutive interval of integers. For every integer t, let $\mathscr{P}_t^n$ denote the collection of all consecutive partitions of $\{1, \ldots, n\}$ with exactly t parts. We do not demand that all of the parts in a partition in $\mathscr{P}_t$ are nonempty. Simple counting arguments show that for every t, $|\mathscr{P}_t^n| = \binom{n+t-1}{t-1}$.

We will denote by $\mathcal{Z}_{s,t}^p$ the set of all t-tuples $(p_1, p_2, \ldots, p_t)$ of integers such that $\sum_{i \leq t} p_i = p$ and $0 \leq p_i \leq s$ for all i. Clearly $|\mathcal{Z}_{s,t}^p| \leq \binom{p+t-1}{t-1}$, since this counts all the ways of writing p as a sum of t nonnegative integers, without considering the upper bound on each one. For ease of convenience, we summarize the aforementioned in the next definition and the proposition.

Definition 4.9. A *partition* of U is a family $\mathcal{U}_P = \{U_1, U_2, \ldots U_t\}$ of sets over U such that $\forall i \neq j,\ U_i \cap U_j = \emptyset$, and $U = \bigcup_{i \leq t} U_i$. Each of the sets U_i are called the *parts* of the partition. A *consecutive partition* of $\{1, \ldots, n\}$ is a partition $\mathcal{U}_P = \{U_1, U_2, \ldots U_t\}$ of $\{1, \ldots, n\}$ such that for every integer $i \leq t$ and integers $1 \leq x \leq y \leq z$, if $x \in U_i$ and $z \in U_i$, then $y \in U_i$ as well.

PROPOSITION 4.10. *Let $\mathscr{P}_t^n$ denote the collection of all consecutive partitions of $\{1, \ldots, n\}$ with exactly t parts. Let $\mathcal{Z}_{s,t}^p$ be the set of all t-tuples $(p_1, p_2, \ldots, p_t)$ of integers such that $\sum_{i \leq t} p_i = p$ and $0 \leq p_i \leq s$ for all i. Then, for every t, $|\mathscr{P}_t^n| = \binom{n+t-1}{t-1}$ and $|\mathcal{Z}_{s,t}^p| \leq \binom{p+t-1}{t-1}$.*

LEMMA 4.11. *For any p, q, let $s = \lfloor (\log(p+q))^2 \rfloor$ and $t = \lceil \frac{p+q}{s} \rceil$. If there is a construction of n-p-q-separating collections $(\mathcal{F}_p, \chi_p, \chi'_p)$ with*

—size $\zeta(n, p, q)$ and initialization time $\tau_I(n, p, q)$,
—(χ_p, p')-degree $\Delta_{(\chi_p, p')}(n, p, q)$ and (χ'_p, q')-degree $\Delta_{(\chi'_p, q')}(n, p, q)$, and
—query times $Q_{(\chi_p, p')}(n, p, q)$ and $Q_{(\chi'_p, q')}(n, p, q)$,

then there is a construction of n-p-q-separating collection with the following parameters:

—

$$\zeta'(n, p, q) \leq |\mathscr{P}_t^n| \cdot \sum_{(p_1, \ldots, p_t) \in \mathcal{Z}_{s,t}^p} \prod_{i \leq t} \zeta(n, p_i, s - p_i),$$

—

$$\tau'_I(n, p, q) = \mathcal{O}\left(\left(\sum_{\substack{\hat{p} \leq s, p \\ s - \hat{p} \leq q}} \tau_I(n, \hat{p}, s - \hat{p})\right) + \zeta'(n, p, q) \cdot n^{\mathcal{O}(1)}\right),$$

—

$$\Delta'_{(\chi, p')}(n, p, q) \leq \Delta^*_{(\chi, p')}(n, p, q) = |\mathscr{P}_t^n| \cdot |\mathcal{Z}_{s,t}^p| \cdot \max_{\substack{(p_1, \ldots, p_t) \in \mathcal{Z}_{s,t}^p \\ p'_1 \leq p_1, \ldots, p'_t \leq p_t \\ p'_1 + \cdots + p'_t = p'}} \prod_{i \leq t} \Delta_{(\chi_{p_i}, p'_i)}(n, p_i, s - p_i),$$

—

$$Q'_{(\chi,p')}(n,p,q) = \mathcal{O}\left(\Delta^*_{(\chi,p')}(n,p,q)\cdot n^{\mathcal{O}(1)} + |\mathscr{P}^n_t|\cdot|\mathcal{Z}^p_{s,t}|\cdot t\cdot\left(\max_{\substack{\hat{p}'\le\hat{p}\le s\\ \hat{p}-\hat{p}'\le p-p'\\ s-\hat{p}\le q}} Q_{(\chi_{\hat{p}},\hat{p}')}(n,\hat{p},s-\hat{p})\right)\right),$$

—

$$\Delta'_{(\chi',q')}(n,p,q)\le\Delta^*_{(\chi',q')}(n,p,q) = |\mathscr{P}^n_t|\cdot|\mathcal{Z}^p_{s,t}|\cdot\max_{\substack{(p_1,\ldots,p_t)\in\mathcal{Z}^p_{s,t}\\ q'_1\le s-p_1,\ldots,q'_t\le s-p_t\\ q'_1+\cdots+q'_t=q'}}\prod_{i\le t}\Delta_{(\chi'_{p_i},q'_i)}(n,p_i,s-p_i),$$

—

$$Q'_{(\chi',q')}(n,p,q) = \mathcal{O}\left(\Delta^*_{(\chi',q')}(n,p,q)\cdot n^{\mathcal{O}(1)} + |\mathscr{P}^n_t|\cdot|\mathcal{Z}^p_{s,t}|\cdot t\cdot\left(\max_{\substack{\hat{q}'\le\hat{q}\le s\\ \hat{q}-\hat{q}'\le q-q'\\ s-\hat{q}\le p}} Q_{(\chi'_{s-\hat{q}},\hat{q}')}(n,s-\hat{q},\hat{q})\right)\right).$$

Proof. Set $s=\lfloor(\log(p+q))^2\rfloor$ and $t=\lceil\frac{p+q}{s}\rceil$. We will give a construction of n-p-q-separating collections with initialization time, query time, size, and degree within the claimed bounds earlier. In this construction, we will use the given construction as a black box. We may assume without loss of generality that $U=\{1,\ldots,n\}$. Our algorithm first runs for every $\hat{p}$, $0\le\hat{p}\le s$, $\hat{p}\le p$, $s-\hat{p}\le q$, and initializes n-$\hat{p}$-$(s-\hat{p})$-separating collections,

$$(\mathcal{F}_{\hat{p}},\chi_{\hat{p}},\chi'_{\hat{p}}).$$

These will be the building blocks of our construction.

We need to define a few operations on families of sets. For families of sets $\mathcal{A}$, $\mathcal{B}$ over U and subset $U'\subseteq U$, we define

$$\begin{aligned}\mathcal{A}\sqcap U' &= \{A\cap U' : A\in\mathcal{A}\}\\ \mathcal{A}\circ\mathcal{B} &= \{A\cup B : A\in\mathcal{A}\wedge B\in\mathcal{B}\}.\end{aligned}$$

We now define $\mathcal{F}$ as follows:

$$\mathcal{F} = \bigcup_{\substack{\{U_1,\ldots,U_t\}\in\mathscr{P}^n_t\\ (p_1,\ldots,p_t)\in\mathcal{Z}^p_{s,t}\text{ such that}\\ \forall i: s-p_i\le q}} (\mathcal{F}_{p_1}\sqcap U_1)\circ(\mathcal{F}_{p_2}\sqcap U_2)\circ\ldots\circ(\mathcal{F}_{p_t}\sqcap U_t). \tag{3}$$

It follows directly from the definition of $\mathcal{F}$ that $|\mathcal{F}|$ is within the claimed bound for $\zeta'(n,p,q)$. For the initialization time, the algorithm spends $\mathcal{O}(\sum_{\substack{\hat{p}\le s,p\\ s-\hat{p}\le q}}\tau_I(n,\hat{p},s-\hat{p}))$ time to initialize the constructions of the n-$\hat{p}$-$(s-\hat{p})$-separating collections for all $\hat{p}\le s$ such that $\hat{p}\le p$ and $s-\hat{p}\le q$ together. Now the algorithm can output the entries of $\mathcal{F}$ one set at a time by using Equation (3), spending $n^{\mathcal{O}(1)}$ time per output set. Hence, the time bound for $\tau'_I(n,p,q)$ follows.

For every set $A\in\bigcup_{p'\le p}\binom{U}{p'}$, we define $\chi(A)$ as follows:

$$\chi(A) = \bigcup_{\substack{\{U_1,\ldots,U_t\}\in\mathscr{P}^n_t\\ (p_1,\ldots,p_t)\in\mathcal{Z}^p_{s,t}\text{ such that}\\ \forall U_i: |U_i\cap A|\le p_i, s-p_i\le q}} \Big[(\chi_{p_1}(A\cap U_1)\sqcap U_1)\circ(\chi_{p_2}(A\cap U_2)\sqcap U_2)\circ\ldots \tag{4}$$

$$\ldots\circ(\chi_{p_t}(A\cap U_t)\sqcap U_t)\Big].$$

Now we show that $\chi(A) \subseteq \mathcal{F}$. From the definition of n-p_i-$(s-p_i)$-separating collections $(\mathcal{F}_{p_i}, \chi_{p_i}, \chi'_{p_i})$, each family $\chi_{p_i}(A \cap U_i)$ in Equation (4) is a subset of $\mathcal{F}_{p_i}$. This implies that $\chi_{p_i}(A \cap U_i) \sqcap U_i \subseteq \mathcal{F}_{p_i} \sqcap U_i$. Hence, $\chi(A) \subseteq \mathcal{F}$. Similarly, we can define $\chi'(B)$ for any $B \in \bigcup_{q' \leq q} \binom{U}{q'}$ as

$$\chi'(B) = \bigcup_{\substack{\{U_1,\ldots,U_t\} \in \mathscr{P}_t^n \\ (p_1,\ldots,p_t) \in \mathcal{Z}_{s,t}^p \text{ such that} \\ \forall U_i : |U_i \cap B| \leq s - p_i \leq q}} \Big[(\chi'_{p_1}(B \cap U_1) \sqcap U_1) \circ (\chi'_{p_2}(B \cap U_2) \sqcap U_2) \circ \cdots \\ \cdots \circ (\chi'_{p_t}(B \cap U_t) \sqcap U_t) \Big]. \tag{5}$$

Similar to the proof of $\chi(A) \subseteq \mathcal{F}$, we can show that $\chi'(B) \subseteq \mathcal{F}$. It follows directly from the definition of $\chi(A)$ and $\chi'(B)$ that $|\chi(A)|$ and $|\chi'(B)|$ are within the claimed bound for $\Delta'_{(\chi,p')}(n,p,q)$ and $\Delta'_{(\chi',q')}(n,p,q)$, respectively. We now describe how queries $\chi(A)$ can be answered and analyze how much time it takes. Given A, we will compute $\chi(A)$ using Equation (4). Let $|A| = p'$. For each $\{U_1,\ldots,U_t\} \in \mathscr{P}_t^n$ and $(p_1,\ldots,p_t) \in \mathcal{Z}_{s,t}^p$ such that $p'_i = |U_i \cap A| \leq p_i, s - p_i \leq q$ for all $i \leq t$, we proceed as follows. First, we compute $\chi_{p_i}(A \cap U_i)$ for each $i \leq t$, spending in total $\mathcal{O}(\sum_{i \leq t} Q_{(\chi_{p_i},p'_i)}(n,p_i,s-p_i))$ time. Now we add each set in

$$(\chi_{p_1}(A \cap U_1) \sqcap U_1) \circ (\chi_{p_2}(A \cap U_2) \sqcap U_2) \circ \ldots \circ (\chi_{p_t}(A \cap U_t) \sqcap U_t)$$

to $\chi(A)$, spending $n^{\mathcal{O}(1)}$ time per set, yielding the following bound:

$$Q'_{(\chi,p')}(n,p,q) \leq \mathcal{O}\left(\Delta^*_{(\chi,p')}(n,p,q) \cdot n^{\mathcal{O}(1)} + \sum_{\substack{\{U_1,\ldots,U_t\} \in \mathscr{P}_t \\ (p_1,\ldots,p_t) \in \mathcal{Z}_{s,t}^p \text{ such that} \\ \forall U_i : p'_i = |U_i \cap A| \leq p_i, s-p_i \leq q}} \left[\sum_{i \leq t} Q_{(\chi_{p_i},p'_i)}(n,p_i,s-p_i) \right] \right)$$

$$\leq \mathcal{O}\left(\Delta^*_{(\chi,p')}(n,p,q) \cdot n^{\mathcal{O}(1)} + |\mathscr{P}_t^n| \cdot |\mathcal{Z}_{s,t}^p| \cdot \max_{\substack{(p_1,\ldots,p_t) \in \mathcal{Z}_{s,t}^p \\ p'_1 \leq p_1, \cdots, p'_t \leq p_t \text{ such that} \\ p'_1 + \cdots + p'_t = p', \forall i : s - p_i \leq q}} \left(\sum_{i \leq t} Q_{(\chi_{p_i},p'_i)}(n,p_i,s-p_i) \right) \right)$$

$$\leq \mathcal{O}\left(\Delta^*_{(\chi,p')}(n,p,q) \cdot n^{\mathcal{O}(1)} + |\mathscr{P}_t^n| \cdot |\mathcal{Z}_{s,t}^p| \cdot t \cdot \max_{\substack{(p_1,\ldots,p_t) \in \mathcal{Z}_{s,t}^p \\ p'_1 \leq p_1, \cdots, p'_t \leq p_t \text{ such that} \\ p'_1 + \cdots + p'_t = p', \forall i : s - p_i \leq q}} \left(Q_{(\chi_{p_i},p'_i)}(n,p_i,s-p_i) \right) \right)$$

$$\leq \mathcal{O}\left(\Delta^*_{(\chi,p')}(n,p,q) \cdot n^{\mathcal{O}(1)} + |\mathscr{P}_t^n| \cdot |\mathcal{Z}_{s,t}^p| \cdot t \cdot \left(\max_{\substack{\hat{p}' \leq \hat{p} \leq s \\ \hat{p} - \hat{p}' \leq p - p' \\ s - \hat{p} \leq q}} Q_{(\chi_{\hat{p}},\hat{p}')}(n,\hat{p},s-\hat{p}) \right) \right).$$

For any $(p_1,\ldots,p_t) \in \mathcal{Z}_{s,t}^p$ and $p'_1 \leq p_1, \ldots, p'_t \leq p_t$ such that $\sum_{i=1}^t p'_i = p'$, we have that $\sum_{i=1}^t p_i - p'_i = p - p'$ and so $p_i - p'_i \leq p - p'$ for all i. This shows the correctness of the last inequality in the previous query time analysis.

By doing a similar analysis, we get the required bound for $Q'_{(\chi',q')}$. We now need to argue that $(\mathcal{F}, \chi, \chi')$ is in fact an n-p-q-separating collection. For any r, consider pairwise disjoint sets $A_1 \in \binom{U}{b_1}, \ldots, A_r \in \binom{U}{b_r}$ and $B \in \binom{U}{q}$ such that $b_1 + \cdots + b_r = p$. Let $A = A_1 \cup \cdots \cup A_r$. There exists a consecutive partition $\{U_1, \ldots, U_t\} \in \mathscr{P}_t^n$ of U such that for every $i \leq t$, we have that $|(A \cup B) \cap U_i| \leq \lceil \frac{p+q}{t} \rceil = s$. For each $i \leq t$, set $p_i = |A \cap U_i|$ and $q_i = |B \cap U_i| = s - p_i$. Note that $p_i \leq p$ and $q_i \leq q$ for all i. For every $i \leq t$, the tuple $(\mathcal{F}_{p_i}, \chi_{p_i}, \chi'_{p_i})$ forms an n-p_i-q_i-separating collection. Hence, there exists a $F_i \in \chi_{p_i}(A_1 \cap U_i) \cap \cdots \cap \chi_{p_i}(A_r \cap U_i) \cap \chi'_{p_i}(B \cap U_i)$ because $|A_1 \cap U_i| + \cdots + |A_r \cap U_i| = p_i$, $|B \cap U_i| = q_i$ and $(\mathcal{F}_{p_i}, \chi_{p_i}, \chi'_{p_i})$ is an n-p_i-q_i-separating collection. That is, $F_i \in \chi_{p_i}(A_j \cap U_i)$ for all $j \leq r$ and $F_i \in \chi'_{p_i}(B \cap U_i)$. Let $F = \bigcup_{i \leq t} F_i \cap U_i$. By construction of χ and χ', $F \in \chi(A_j)$ for all $j \leq r$ and $F \in \chi'(B)$. Hence, $F \in \chi(A_1) \cap \cdots \cap \chi(A_r) \cap \chi'(B)$. This completes the proof. □

Now we are ready to prove Lemma 4.4. We restate the lemma for easiness of presentation.

Lemma 4.4. *Given $0 < x < 1$, there is a construction of n-p-q-separating collection with the following parameters:*

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

Proof. We first explain a brute-force construction of n-p-q-separating collection when the value of x is close to 0 or close to 1. These are discussed in Cases 1 and 2 and the result for all other values of x is explained in Case 3. Let U be the universe.

Case 1: $x \leq \frac{1}{n}$. In this case, the algorithm will output all subsets of size p of the universe as the family $\mathcal{F}$ of sets in the n-p-q-separating collection. That is, $\mathcal{F} = \{F \subseteq U \mid |F| = p\}$. We define χ and χ' as follows. For any $A \in \bigcup_{p' \leq p} \binom{U}{p'}$, $\chi(A) = \{F \in \mathcal{F} \mid A \subseteq F\}$. For any $B \in \bigcup_{q' \leq q} \binom{U}{q'}$, $\chi'(B) = \{F \in \mathcal{F} \mid B \cap F = \emptyset\}$. It is easy to see that $(\mathcal{F}, \chi, \chi')$ is an n-p-q-separating collection. Note that $|\mathcal{F}| = \binom{n}{p} \leq n^p$. Since $n \leq \frac{1}{x}$, the size of the n-p-q-separating collection is upper bounded by the claimed bound. Since we can list all the elements in $\mathcal{F}$ in n^p time, the initialization time is upper bounded by the claimed bound. For any $A \subseteq U$, $|A| = p'$, the cardinality of $\chi(A)$ is exactly equal to $\binom{n}{p-p'}$, which is upper bounded by $\frac{1}{x^{p-p'}}$. Thus, the (χ, p')-degree and (χ, p')-query time is bounded by the claimed bound. For any $B \subseteq U$, $|B| = q'$, the cardinality of $\chi'(B)$ is at most $|\mathcal{F}|$, which is upper bounded by $\frac{1}{x^p}$. Thus, the (χ', q')-degree and (χ', q')-query time is bounded by the claimed bound.

Case 2: $1 - x \leq \frac{1}{n}$. In this case, the algorithm will output all subsets of size $n - q$ of the universe as the family $\mathcal{F}$ of sets in the n-p-q-separating collection. That is, $\mathcal{F} = \{F \subseteq U \mid |F| = n - q\}$. We define χ and χ' as follows. For any $A \in \bigcup_{p' \leq p} \binom{U}{p'}$, $\chi(A) = \{F \in \mathcal{F} \mid A \subseteq F\}$. For any $B \in \bigcup_{q' \leq q} \binom{U}{q'}$, $\chi'(B) = \{F \in \mathcal{F} \mid B \cap F = \emptyset\}$. It is easy to

see that $(\mathcal{F}, \chi, \chi')$ is an n-p-q-separating collection. Note that $|\mathcal{F}| = \binom{n}{n-q} \leq n^q$. Since $n \leq \frac{1}{1-x}$, the size of the n-p-q-separating collection is upper bounded by the claimed bound. Since we can list all the elements in $\mathcal{F}$ in n^q time, the initialization time is upper bounded by the claimed bound. For any $A \subseteq U$, $|A| = p'$, the cardinality of $\chi(A)$ is at most $|\mathcal{F}|$, which is upper bounded by $\frac{1}{(1-x)^q}$. Thus, the (χ, p')-degree and (χ, p')-query time is bounded by the claimed bound. For any $B \subseteq U$, $|B| = q'$, the cardinality of $\chi'(B)$ is exactly equal to $\binom{n}{q-q'}$, which is upper bounded by $\frac{1}{(1-x)^{q-q'}}$. Thus, the (χ', q')-degree and (χ', q')-query time is bounded by the claimed bound.

Case 3: $x, 1-x > \frac{1}{n}$. The structure of the proof in this case is as follows. We first create a collection using Lemma 4.5. Then we apply Lemma 4.8 and obtain another construction. From here onward we keep applying Lemma 4.11 and Lemma 4.8 in phases until we achieve the required bounds on size, degree, query, and initialization time.

We first apply Lemma 4.5 and get a construction of n-p-q-separating collections with the following parameters:

—size, $\zeta^1(n, p, q) = \mathcal{O}(\frac{1}{x^p(1-x)^q} \cdot (p^2 + q^2 + 1) \log n)$
—initialization time, $\tau_I^1(n, p, q) = \mathcal{O}(\binom{2^n}{\zeta(n,p,q)} \cdot \frac{1}{x^p(1-x)^q} \cdot n^{\mathcal{O}(p+q)})$
—(χ_1, p')-degree for $p' \leq p$, $\Delta^1_{(\chi_1,p')}(n, p, q) = \mathcal{O}(\frac{1}{x^{p-p'}} \cdot \frac{(p^2+q^2+1)}{(1-x)^q} \cdot \log n)$
—(χ_1, p')-query time $Q^1_{(\chi_1,p')}(n, p, q) = \mathcal{O}(\frac{1}{x^p(1-x)^q} \cdot n^{\mathcal{O}(1)}) = \mathcal{O}(2^n n^{\mathcal{O}(1)})$
—(χ'_1, q')-degree for $q' \leq q$, $\Delta^1_{(\chi'_1,q')}(n, p, q) = \mathcal{O}(\frac{1}{x^p(1-x)^{q-q'}} \cdot (p^2 + q^2 + 1) \cdot \log n)$
—(χ'_1, q')-query time, $Q^1_{(\chi'_1,q')}(n, p, q) = \mathcal{O}(\frac{1}{x^p(1-x)^q} \cdot n^{\mathcal{O}(1)}) = \mathcal{O}(2^n n^{\mathcal{O}(1)})$

We apply Lemma 4.8 to this construction to get a new construction with the following parameters:

—size, $\zeta^2(n, p, q) = \mathcal{O}(\frac{1}{x^p(1-x)^q} \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$
—initialization time,

$$\begin{aligned}
\tau_I^2(n, p, q) &= \mathcal{O}\big(\tau_I^1\big((p+q)^2, p, q\big) + \zeta^1\big((p+q)^2, p, q\big) \cdot (p+q)^{\mathcal{O}(1)} \cdot n \log n\big) \\
&= \mathcal{O}\left(\frac{2^{2^{(p+q)^2}}}{x^p(1-x)^q} \cdot (p+q)^{\mathcal{O}(p+q)} + \left(\frac{1}{x^p(1-x)^q} \cdot (p+q)^{\mathcal{O}(1)} \cdot n \log n\right)\right) \\
&= \mathcal{O}\left(\frac{(p+q)^{\mathcal{O}(p+q)}}{x^p(1-x)^q}\left(2^{2^{(p+q)^2}} + n \log n\right)\right)
\end{aligned}$$

—(χ_2, p')-degree, $\Delta^2_{(\chi_2,p')}(n, p, q) = \mathcal{O}(\frac{1}{x^{p-p'}(1-x)^q} \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$
—(χ_2, p')-query time, $Q^2_{(\chi_2,p')}(n, p, q) = \mathcal{O}((2^{(p+q)^2} + \frac{1}{x^{p-p'}(1-x)^q})(p+q)^{\mathcal{O}(1)} \cdot \log n)$
—(χ'_2, q')-degree, $\Delta^2_{(\chi'_2,q')}(n, p, q) = \mathcal{O}(\frac{1}{x^p(1-x)^{q-q'}} \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$
—(χ_2, q')-query time, $Q^2_{(\chi'_2,q')}(n, p, q) = \mathcal{O}((2^{(p+q)^2} + \frac{1}{x^p(1-x)^{q-q'}})(p+q)^{\mathcal{O}(1)} \cdot \log n)$

We apply Lemma 4.11 to this construction. Recall that in Lemma 4.11 we set $s = \lfloor(\log(p+q))^2\rfloor$ and $t = \lceil\frac{p+q}{s}\rceil$:

$$\begin{aligned}
\zeta^3(n,p,q) &\le |\mathscr{P}_t^n| \cdot \sum_{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p} \prod_{i\le t} \zeta^2(n,p_i,s-p_i) \\
&\le n^{\mathcal{O}(t)} \cdot |\mathcal{Z}_{s,t}^p| \cdot \max_{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p} \prod_{i\le t} \zeta^2(n,p_i,s-p_i) \\
&\le n^{\mathcal{O}(t)} \cdot (p+q)^{\mathcal{O}(t)} \cdot \frac{1}{x^p(1-x)^{q+s}} \cdot s^{\mathcal{O}(t)} \cdot (\log n)^{\mathcal{O}(t)} \\
&\le n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})} \cdot \frac{1}{x^p(1-x)^q} \qquad \left(\text{Because } \left(\frac{1}{1-x}\right)^s \le n^s \le n^{\mathcal{O}(t)}\right)
\end{aligned}$$

$$\begin{aligned}
\tau_I^3(n,p,q) &= \mathcal{O}\left(\left(\sum_{\substack{\hat{p}\le s,p\\ s-\hat{p}\le q}} \tau_I^2(n,\hat{p},s-\hat{p})\right) + \zeta^3(n,p,q)\cdot n^{\mathcal{O}(1)}\right) \\
&= \mathcal{O}\left(\left(\sum_{\substack{\hat{p}\le s,p\\ s-\hat{p}\le q}} \frac{s^{\mathcal{O}(s)}}{x^{\hat{p}}(1-x)^{s-\hat{p}}}\left(2^{2^{s^2}} + n\log n\right)\right) + \zeta^3(n,p,q)\cdot n^{\mathcal{O}(1)}\right) \\
&= \mathcal{O}\left(\frac{(\log(p+q))^{\mathcal{O}(\log^2(p+q))}}{x^p(1-x)^q}\left(2^{2^{\log^4(p+q)}} + n\log n\right) + n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot\frac{1}{x^p(1-x)^q}\right)
\end{aligned}$$

$$\begin{aligned}
\Delta^3_{(\chi_3,p')}(n,p,q) &\le \Delta^{*3}_{(\chi_3,p')}(n,p,q) \\
&= |\mathscr{P}_t^n| \cdot |\mathcal{Z}_{s,t}^p| \cdot \max_{\substack{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p\\ p_1'\le p_1,\ldots,p_t'\le p_t\\ p_1'+\cdots+p_t'=p'}} \prod_{i\le t} \Delta^2_{(\chi,p')}(n,p_i,s-p_i) \\
&\le n^{\mathcal{O}(t)} \cdot (p+q)^{\mathcal{O}(t)} \cdot \frac{1}{x^{p-p'}(1-x)^{q+s}} \cdot s^{\mathcal{O}(t)} \cdot (\log n)^{\mathcal{O}(t)} \\
&\le n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})} \cdot \frac{1}{x^{p-p'}(1-x)^q} \qquad \left(\text{Because } \left(\frac{1}{1-x}\right)^s \in n^{\mathcal{O}(t)}\right) \\
\Delta^3_{(\chi_3',q')}(n,p,q) &\le \Delta^{*3}_{(\chi_3',q')}(n,p,q) \\
&= |\mathscr{P}_t^n| \cdot |\mathcal{Z}_{s,t}^p| \cdot \max_{\substack{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p\\ q_1'\le s-p_1,\ldots,q_t'\le s-q_t\\ q_1'+\cdots+q_t'=q'}} \prod_{i\le t} \Delta^2_{(\chi',q_i')}(n,p_i,s-p_i) \\
&\le n^{\mathcal{O}(t)} \cdot (p+q)^{\mathcal{O}(t)} \cdot \frac{1}{x^p(1-x)^{q+s-q'}} \cdot s^{\mathcal{O}(t)} \cdot (\log n)^{\mathcal{O}(t)} \\
&\le n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})} \cdot \frac{1}{x^p(1-x)^{q-q'}} \qquad \left(\text{Because } \left(\frac{1}{1-x}\right)^s \in n^{\mathcal{O}(t)}\right)
\end{aligned}$$

$$
\begin{aligned}
Q^3_{(\chi_3,p')}(n,p,q) &\le \mathcal{O}\left(\Delta^{*3}_{(\chi_3,p')}(n,p,q)\cdot n^{\mathcal{O}(1)} + |\mathscr{P}^n_t|\cdot|\mathcal{Z}^p_{s,t}|\cdot t\cdot \max_{\substack{\hat{p}'\le\hat{p}\le s\\ \hat{p}-\hat{p}'\le p-p'\\ s-\hat{p}\le q}} Q^2_{(\chi_2,\hat{p}')}(n,\hat{p},s-\hat{p})\right)\\
&\le \mathcal{O}\left(\Delta^{*3}_{(\chi_3,p')}(n,p,q)\cdot n^{\mathcal{O}(1)} + \right.\\
&\qquad\left. n^{\mathcal{O}(t)}\max_{\substack{\hat{p}'\le\hat{p}\le s\\ \hat{p}-\hat{p}'\le p-p'\\ s-\hat{p}\le q}}\left(2^{s^2}+\frac{1}{x^{\hat{p}-\hat{p}'}(1-x)^{s-\hat{p}}}\right)s^{\mathcal{O}(1)}\log n\right)\\
&\le \mathcal{O}\left(\frac{n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}}{x^{p-p'}(1-x)^q}+n^{\mathcal{O}(t)}\cdot s^{\mathcal{O}(1)}\cdot\log n\left(2^{s^2}+\frac{1}{x^{p-p'}(1-x)^q}\right)\right)\\
&\le \mathcal{O}\left(\frac{n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}}{x^{p-p'}(1-x)^q}\right)
\end{aligned}
$$

In a similar way, we can bound $Q^3_{(\chi_3',q')}$ as

$$
Q^3_{(\chi_3',q')}(n,p,q) \le \mathcal{O}\left(\frac{n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}}{x^p(1-x)^{q-q'}}\right).
$$

We apply Lemma 4.8 to this construction to get a new construction with the following parameters:

—size, $\zeta^4(n,p,q)\le 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot\frac{1}{x^p(1-x)^q}\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n$
—initialization time,

$$
\begin{aligned}
\tau^4_I(n,p,q) &\le \mathcal{O}\big(\tau^3_I\big((p+q)^2,p,q\big)+\zeta^3\big((p+q)^2,p,q\big)\cdot(p+q)^{\mathcal{O}(1)}\cdot n\log n\big)\\
&\le 2^{2^{\log^4(p+q)}}\cdot\frac{(\log(p+q))^{\mathcal{O}(\log^2(p+q))}}{x^p(1-x)^q}+\frac{2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}}{x^p(1-x)^q}\cdot(p+q)^{\mathcal{O}(1)}n\log n
\end{aligned}
$$

—(χ_4,p')-degree,

$$
\begin{aligned}
\Delta^4_{(\chi_4,p')}(n,p,q) &\le \Delta^3_{(\chi_3,p')}\big((p+q)^2,p,q\big)\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\\
&\le \frac{2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}}{x^{p-p'}(1-x)^q}\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n
\end{aligned}
$$

—(χ_4',q')-degree,

$$
\begin{aligned}
\Delta^4_{(\chi_4',q')}(n,p,q) &\le \Delta^3_{(\chi_3',q')}\big((p+q)^2,p,q\big)\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\\
&\le \frac{2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}}{x^p(1-x)^{q-q'}}\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n
\end{aligned}
$$

—(χ_4, p')-query time,

$$\begin{aligned} Q^4_{(\chi_4,p')}(n,p,q) &\le \mathcal{O}\big(\big(Q^3_{(\chi_3,p')}\big((p+q)^2,p,q\big)+\Delta^3_{(\chi_3,p')}\big((p+q)^2,p,q\big)\big)\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\big)\\ &\le \frac{2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}}{x^{p-p'}(1-x)^q}\cdot(p+q)^{\mathcal{O}(1)}\log n \end{aligned}$$

—(χ_4', q')-query time,

$$Q^4_{(\chi_4',q')}(n,p,q) \le \frac{2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}}{x^p(1-x)^{q-q'}}\cdot(p+q)^{\mathcal{O}(1)}\log n$$

We apply Lemma 4.11 to this construction by setting $s=\lfloor(\log(p+q))^2\rfloor$ and $t=\lceil\frac{p+q}{s}\rceil$:

—size,

$$\begin{aligned} \zeta^5(n,p,q) &\le |\mathscr{P}^n_t|\cdot\sum_{(p_1,\ldots,p_t)\in\mathcal{Z}^p_{s,t}}\prod_{i\le t}\zeta^4(n,p_i,s-p_i)\\ &\le n^{\mathcal{O}(t)}\cdot(p+q)^{\mathcal{O}(t)}\cdot s^{\mathcal{O}(t)}\cdot 2^{\mathcal{O}(\frac{st}{\log s})}\cdot(\log n)^{\mathcal{O}(t)}\cdot\frac{1}{x^p(1-x)^{q+s}}\\ &\le n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\frac{1}{x^p(1-x)^q} \qquad \left(\text{Because } \left(\frac{1}{1-x}\right)^s\in n^{\mathcal{O}(t)}\right) \end{aligned}$$

—initialization time,

$$\begin{aligned} \tau^5_I(n,p,q) &\le \mathcal{O}\left(\left(\sum_{\substack{\hat{p}\le s,p\\ s-\hat{p}\le q}}\tau^4_I(n,\hat{p},s-\hat{p})\right)+\zeta^5(n,p,q)\cdot n^{\mathcal{O}(1)}\right)\\ &\le \mathcal{O}\left(s\frac{2^{2^{\log^4 s}}\cdot(\log s)^{\mathcal{O}(\log^2 s)}}{x^p(1-x)^q}+\frac{2^{\mathcal{O}(\frac{s}{\log s})}}{x^p(1-x)^q}\cdot n\log n+\right.\\ &\qquad\left. n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot\frac{2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}}{x^p(1-x)^q}\right)\\ &\le \mathcal{O}\left(s\frac{2^{2^{\log^4 s}}\cdot(\log s)^{\mathcal{O}(\log^2 s)}}{x^p(1-x)^q}+n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot\frac{2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}}{x^p(1-x)^q}\right)\\ &\le \mathcal{O}\left(\frac{2^{2^{\log^4 s}}\cdot(s)^{\mathcal{O}(s)}}{x^p(1-x)^q}+n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot\frac{2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}}{x^p(1-x)^q}\right)\\ &\le \mathcal{O}\left(\frac{2^{2^{(2\log\log(p+q))^4}}\cdot(\log(p+q))^{\mathcal{O}((\log(p+q))^2)}}{x^p(1-x)^q}+n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot\frac{2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}}{x^p(1-x)^q}\right)\\ &\le \mathcal{O}\left(n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot\frac{2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}}{x^p(1-x)^q}\right)\\ &\qquad\left(\text{Because } 2^{2^{(2\log\log(p+q))^4}},(\log(p+q))^{\mathcal{O}(\log^2(p+q))}\le 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\right) \end{aligned}$$

—(χ_5, p')-degree,

$$\begin{aligned}\Delta^5_{(\chi_5,p')}(n,p,q) &\le \Delta^{*5}_{(\chi_5,p')}(n,p,q)\\ &= |\mathscr{P}^n_t| \cdot |\mathcal{Z}^p_{s,t}| \cdot \max_{\substack{(p_1,\ldots,p_t)\in\mathcal{Z}^p_{s,t}\\ p'_1\le p_1,\ldots,p'_t\le p_t\\ p'_1+\cdots+p'_t=p'}} \prod_{i\le t} \Delta^4_{(\chi_4,p'_i)}(n,p_i,s-p_i)\\ &\le n^{\mathcal{O}(t)}\cdot(p+q)^{\mathcal{O}(t)}\cdot\frac{2^{\mathcal{O}(\frac{st}{\log s})}}{x^{p-p'}(1-x)^{q+s}}\cdot s^{\mathcal{O}(t)}\cdot(\log n)^{\mathcal{O}(t)}\\ &\le n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot\frac{1}{x^{p-p'}(1-x)^q}\\ &\qquad\qquad \left(\text{Because } \left(\frac{1}{1-x}\right)^s \in n^{\mathcal{O}(t)}\right)\end{aligned}$$

—(χ'_5, q')-degree,

$$\begin{aligned}\Delta^5_{(\chi'_5,q')}(n,p,q) &\le \Delta^{*5}_{(\chi'_5,q')}(n,p,q)\\ &\le n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot\frac{1}{x^p(1-x)^{q-q'}}\end{aligned}$$

—(χ_5, p')-query time,

$$\begin{aligned}Q^5_{(\chi_5,p')}(n,p,q) &\le \mathcal{O}\left(\Delta^{*5}_{(\chi_5,p')}(n,p,q)\cdot n^{\mathcal{O}(1)} + |\mathscr{P}^n_t|\cdot|\mathcal{Z}^p_{s,t}|\cdot \max_{\substack{\hat{p}'\le\hat{p}\le s\\ \hat{p}-\hat{p}'\le p-p'\\ s-\hat{p}\le q}} Q^4_{(\chi_4,\hat{p}')}(n,\hat{p},s-\hat{p})\right)\\ &\le n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot\frac{1}{x^{p-p'}(1-x)^q}\end{aligned}$$

—(χ'_5, q')-query time,

$$Q^5_{(\chi'_5,q')}(n,p,q) \le n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot\frac{1}{x^p(1-x)^{q-q'}}$$

We apply Lemma 4.8 to this construction to get a new construction with the following parameters:

—size,

$$\begin{aligned}\zeta(n,p,q) &\le \zeta^5\big((p+q)^2,p,q\big)\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\\ &\le 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot\frac{1}{x^p(1-x)^q}\cdot(p+q)^{\mathcal{O}(1)}\log n\end{aligned}$$

—initialization time,

$$\begin{aligned}\tau_I(n,p,q) &\le \mathcal{O}\big(\tau^5_I\big((p+q)^2,p,q\big)+\zeta^5\big((p+q)^2,p,q\big)\cdot(p+q)^{\mathcal{O}(1)}\cdot n\log n\big)\\ &= \mathcal{O}\left(2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot\frac{1}{x^p(1-x)^q}\cdot(p+q)^{\mathcal{O}(1)}n\log n\right)\end{aligned}$$

—(χ, p')-degree,

$$\begin{aligned}\Delta_{(\chi,p')}(n,p,q) &\leq \Delta^5_{(\chi_5,p')}\big((p+q)^2,p,q\big)\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\\ &\leq \mathcal{O}\left(2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot\frac{1}{x^{p-p'}(1-x)^q}\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\right)\end{aligned}$$

—(χ, p')-query time,

$$\begin{aligned}Q_{(\chi,p')}(n,p,q) &\leq \mathcal{O}\big(\big(Q^5_{(\chi_5,p')}\big((p+q)^2,p,q\big)+\Delta^5_{(\chi_5,p')}\big((p+q)^2,p,q\big)\big)\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\big)\\ &\leq \mathcal{O}\left(2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot\frac{1}{x^{p-p'}(1-x)^q}\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\right)\end{aligned}$$

—(χ', q')-degree,

$$\begin{aligned}\Delta_{(\chi',q')}(n,p,q) &= \Delta^5_{(\chi'_5,q')}\big((p+q)^2,p,q\big)\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\\ &\leq \mathcal{O}\left(2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot\frac{1}{x^{p}(1-x)^{q-q'}}\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\right)\end{aligned}$$

—(χ', q')-query time,

$$\begin{aligned}Q_{(\chi',q')}(n,p,q) &= \mathcal{O}\big(\big(Q^5_{(\chi'_5,q')}\big((p+q)^2,p,q\big)+\Delta^5_{(\chi'_5,q')}\big((p+q)^2,p,q\big)\big)\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\big)\\ &\leq \mathcal{O}\left(2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot\frac{1}{x^{p}(1-x)^{q-q'}}\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n\right)\end{aligned}$$

The final construction satisfies all the claimed bounds. This concludes the proof. □

LEMMA 4.12. *There is an algorithm that, given a p-family $\mathcal{A}$ of sets over a universe U of size n, an integer q, a $0 < x < 1$, and a nonnegative weight function $w : \mathcal{A} \to \mathbb{N}$ with maximum value at most W, computes in time*

$$\mathcal{O}(x^{-p}(1-x)^{-q}\cdot 2^{o(p+q)}\cdot n\log n + |\mathcal{A}|\cdot\log|\mathcal{A}|\cdot\log W + |\mathcal{A}|\cdot(1-x)^{-q}\cdot 2^{o(p+q)}\cdot\log n)$$

a subfamily $\widehat{\mathcal{A}} \subseteq \mathcal{A}$ such that $|\hat{\mathcal{A}}| \leq x^{-p}(1-x)^{-q}\cdot 2^{o(p+q)}\cdot\log n$ and $\widehat{\mathcal{A}} \subseteq^q_{minrep} \mathcal{A}$ ($\widehat{\mathcal{A}} \subseteq^q_{maxrep} \mathcal{A}$).

PROOF. The algorithm first checks whether $|\mathcal{A}| \leq x^{-p}(1-x)^{-q}\cdot 2^{o(p+q)}\cdot\log n$. If yes, then it outputs $\mathcal{A}$ (as $\widehat{\mathcal{A}}$) and halts. So we assume that $|\mathcal{A}| > x^{-p}(1-x)^{-q}\cdot 2^{o(p+q)}\cdot\log n$. The algorithm starts by constructing a generalized n-p-q-separating collection $(\mathcal{F}, \chi, \chi')$ as guaranteed by Lemma 4.4. If $|\mathcal{A}| \leq |\mathcal{F}|$, the algorithm outputs $\mathcal{A}$ and halts. Otherwise, it builds the set $\hat{\mathcal{A}}$ as follows. Initially, $\hat{\mathcal{A}}$ is equal to $\emptyset$ and all sets in $\mathcal{F}$ are marked as unused. Now we sort the sets in $\mathcal{A}$ in the increasing order of weights, given by $w : \mathcal{A} \to \mathbb{N}$. The algorithm goes through every $A \in \mathcal{A}$ in the sorted order and queries the separating collection to get the set $\chi(A)$. It then looks for a set $F \in \chi(A)$ that is not yet marked as used. The first time such a set F is found, the algorithm marks F as used, inserts A into $\hat{\mathcal{A}}$, and proceeds to the next set in $\mathcal{A}$. If no such set F is found, the algorithm proceeds to the next set in $\mathcal{A}$ without inserting A into $\hat{\mathcal{A}}$.

The size of $\hat{\mathcal{A}}$ is upper bounded by $|\mathcal{F}| \leq x^{-p}(1-x)^{-q}\cdot 2^{o(p+q)}\cdot\log n$ since every time a set is added to $\hat{\mathcal{A}}$, an unused set in $\mathcal{F}$ is marked as used. For the running time analysis, the initialization of $(\mathcal{F}, \chi)$ takes time $x^{-p}(1-x)^{-q}\cdot(p+q)^{\mathcal{O}(1)}\cdot 2^{o(p+q)}\cdot n\log n$. Sorting $\mathcal{A}$ takes $\mathcal{O}(|\mathcal{A}|\cdot\log|\mathcal{A}|\cdot\log W)$ time. For each element $A \in \mathcal{A}$, the algorithm first queries $\chi(A)$, using time $(1-x)^{-q}\cdot 2^{o(p+q)}\cdot(p+q)^{\mathcal{O}(1)}\cdot\log n$. Then it goes through all sets in $\chi(A)$ and checks whether they have already been marked as used, taking time $(1-x)^{-q}\cdot(p+q)^{\mathcal{O}(1)}\cdot 2^{o(p+q)}\cdot\log n$. Thus, in total, the running time for these steps is

bounded by $\mathcal{O}(|\mathcal{A}| \cdot (1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n + |\mathcal{A}| \cdot \log |\mathcal{A}| \cdot \log W)$. Adding the initialization time to this gives the claimed running time.

Finally, we need to argue that $\widehat{\mathcal{A}} \subseteq^q_{minrep} \mathcal{A}$. Consider any set $A \in \mathcal{A}$ and B such that $|B| = q$ and $A \cap B = \emptyset$. If $A \in \hat{\mathcal{A}}$, we are done, so assume that $A \notin \hat{\mathcal{A}}$. Since $(\mathcal{F}, \chi, \chi')$ is an n-p-q-separating collection, we have that there exists $F \in \chi(A) \cap \chi'(B)$, that is, $A \subseteq F$ and $F \cap B = \emptyset$. Since $A \notin \hat{\mathcal{A}}$, we know that F was marked as used when A was considered by the algorithm. When the algorithm marked F as used, it also inserted a set A' into $\hat{\mathcal{A}}$, with the property that $F \in \chi(A')$. Thus, $A' \subseteq F$ and hence $A' \cap B = \emptyset$. Furthermore, A' was considered before A and thus $w(A') \leq w(A)$. But $A' \in \hat{\mathcal{A}}$, completing the proof. □

Next we prove a "faster version of Lemma 4.12" that speeds up the running time to compute the representative families.

Lemma 4.13. *There is an algorithm that, given a p-family $\mathcal{A}$ of sets over a universe U of size n, an integer q, a $0 < x < 1$, and a nonnegative weight function $w : \mathcal{A} \to \mathbb{N}$ with maximum value at most W, computes in time*

$$\mathcal{O}\big((p+q)^{\mathcal{O}(1)} n \log n + |\mathcal{A}| \cdot \log |\mathcal{A}| \cdot \log W + |\mathcal{A}| \cdot (1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n\big)$$

a subfamily $\widehat{\mathcal{A}} \subseteq \mathcal{A}$ such that $|\hat{\mathcal{A}}| \leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n$ and $\widehat{\mathcal{A}} \subseteq^q_{minrep} \mathcal{A}$ ($\widehat{\mathcal{A}} \subseteq^q_{maxrep} \mathcal{A}$).

Proof. The algorithm first checks whether $|\mathcal{A}| \leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n$. If yes, then it outputs $\mathcal{A}$ (as $\widehat{\mathcal{A}}$) and halts. So we assume that $|\mathcal{A}| > x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n$.

We start by constructing a $(p+q)$-perfect family $f_1, \ldots, f_t$ of hash functions from U to $[(p+q)^2]$ with $t = \mathcal{O}((p+q)^{\mathcal{O}(1)} \cdot \log n)$ in time $\mathcal{O}(k^{\mathcal{O}(1)} n \log n)$ using Proposition 4.7. Now we sort the sets in $\mathcal{A}$ in the increasing order of weights, given by $w : \mathcal{A} \to \mathbb{N}$. For every f_j, $1 \leq j \leq t$, we construct a family $\hat{\mathcal{A}}_j$ as follows. The algorithm starts by constructing a generalized $[(p+q)^2]$-p-q-separating collection $(\mathcal{F}_j, \chi_j, \chi'_j)$ as guaranteed by Lemma 4.4. It builds the set $\hat{\mathcal{A}}_j$ as follows. Initially, $\hat{\mathcal{A}}_j$ is equal to $\emptyset$ and all sets in $\mathcal{F}$ are marked as unused. The algorithm goes through every $A \in \mathcal{A}$ in the sorted order and does as follows:

—It first checks whether every element in A gets mapped to distinct integers by f_j. That is, $|\{f_j(a) \mid a \in A\}| = |A|$. If $|\{f_j(a) \mid a \in A\}| < |A|$, then the algorithm proceeds to the next set in $\mathcal{A}$ without inserting A into $\hat{\mathcal{A}}$. Otherwise, we move to the next step.
—It queries the separating collection to get the set $\chi(A)$. It looks for a set $F \in \chi_j(A)$ that is not yet marked as used. The first time such a set F is found, the algorithm marks F as used, inserts A into $\hat{\mathcal{A}}_j$, and proceeds to the next set in $\mathcal{A}$. If no such set F is found, the algorithm proceeds to the next set in $\mathcal{A}$ without inserting A into $\hat{\mathcal{A}}_j$.

Finally, we return $\hat{\mathcal{A}} = \bigcup_{j=1}^{t} \hat{\mathcal{A}}_j$.

The size of $\hat{\mathcal{A}}_j$ is upper bounded by $|\mathcal{F}| \leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log(p+q)$ since every time a set is added to $\hat{\mathcal{A}}$, an unused set in $\mathcal{F}$ is marked as used. Thus, the size of $\hat{\mathcal{A}}$ is upper bounded by $|\mathcal{F}| \leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log(p+q) \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n \leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n$. The running time analysis follows similarly to the one given in Lemma 4.12.

Finally, we need to argue that $\widehat{\mathcal{A}} \subseteq^q_{minrep} \mathcal{A}$. Consider any set $A \in \mathcal{A}$ and B such that $|B| = q$ and $A \cap B = \emptyset$. If $A \in \hat{\mathcal{A}}$, we are done, so assume that $A \notin \hat{\mathcal{A}}$. By the properties of $(p+q)$-perfect family $f_1, \ldots, f_t$ of hash functions from U to $[(p+q)^2]$, there exists an integer $j \in \{1, \ldots, t\}$ such that f_j is injective on $A \cup B$. We focus now on the construction

of $\hat{\mathcal{A}}_j$. Since $(\mathcal{F}_j, \chi_j, \chi'_j)$ is a $[(p+q)^2]$-p-q-separating collection, we have that there exists $F \in \chi_j(A) \cap \chi'_j(B)$, that is, $A \subseteq F$ and $F \cap B = \emptyset$. Since $A \notin \hat{\mathcal{A}}_j$ (as $A \notin \hat{\mathcal{A}}$), we know that F was marked as used when A was considered by the algorithm. When the algorithm marked F as used, it also inserted a set A' into $\hat{\mathcal{A}}$, with the property that $F \in \chi(A')$. Thus, $A' \subseteq F$ and hence $A' \cap B = \emptyset$. Furthermore, A' was considered before A and thus $w(A') \leq w(A)$. But $A' \in \hat{\mathcal{A}}_j \subseteq \hat{\mathcal{A}}$, completing the proof. □

While applying Lemma 4.13, we can reduce the universe size to at most $|\mathcal{A}|p + q$. The next lemma formalizes this.

LEMMA 4.14. *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 nonnegative weight function $w : \mathcal{A} \to \mathbb{N}$ with maximum value at most W, computes in time*

$$\mathcal{O}\big(|\mathcal{A}| \cdot \log|\mathcal{A}| \cdot \log W + |\mathcal{A}| \cdot (1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n\big)$$

a subfamily $\widehat{\mathcal{A}} \subseteq \mathcal{A}$ such that $|\widehat{\mathcal{A}}| \leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log|\mathcal{A}|$ and $\widehat{\mathcal{A}} \subseteq^q_{minrep} \mathcal{A}$ ($\widehat{\mathcal{A}} \subseteq^q_{maxrep} \mathcal{A}$).

PROOF. We first construct a new universe U' as follows. If $n \leq |\mathcal{A}|p + q$, then we set $U' = U$; otherwise, U' will consist of elements from U that are part of any set in $\mathcal{A}$ and q new elements. The universe U' can be constructed in $\mathcal{O}(|\mathcal{A}|p + q)$ time. Also, note that $|U'| \leq |\mathcal{A}|p + q$ and $|U'| \leq n$. Now we claim that a q-representative family $\widehat{\mathcal{A}}$ of $\mathcal{A}$ with respect to the universe U' is also the required representative family over U. Suppose $X \in \mathcal{A}$ and $Y \subseteq U$, $|Y| \leq q$ such that $X \cap Y = \emptyset$. Let $Y' = Y \setminus U'$ and let Y'' be an arbitrary subset of size $|Y'|$ of $U' \setminus U$. Let $Z = (Y \setminus Y') \cup Y''$. It is easy to see that $|Z| = |Y|$ and $X \cap Z = \emptyset$. By the definition of a q-representative family, there exists $\widehat{X} \in \widehat{\mathcal{A}}$ such that $\widehat{X} \cap Z = \emptyset$. Since $Y' \cap \widehat{X} = \emptyset$, we have that $\widehat{X} \cap Y = \emptyset$.

Thus, we apply Lemma 4.13 to compute the q-representative family $\widehat{\mathcal{A}}$ of $\mathcal{A}$ with respect to the universe U' and output it as the desired family. The claimed running time and the size bound on the output representative family follow by substituting the upper bound on $|U'|$ in the bounds coming from Lemma 4.13. □

Finally, we give our main theorem.

THEOREM 4.15. *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 nonnegative weight function $w : \mathcal{A} \to \mathbb{N}$ with maximum value at most W, computes in time*

$$\mathcal{O}\big(|\mathcal{A}| \cdot \log|\mathcal{A}| \cdot \log W + |\mathcal{A}| \cdot (1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n\big)$$

a subfamily $\widehat{\mathcal{A}} \subseteq \mathcal{A}$ such that $|\hat{\mathcal{A}}| \leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)}$ and $\widehat{\mathcal{A}} \subseteq^q_{minrep} \mathcal{A}$ ($\widehat{\mathcal{A}} \subseteq^q_{maxrep} \mathcal{A}$).

PROOF. Let $\mathcal{A} = \mathcal{A}_1$. We compute a sequence of representative families

$$\mathcal{A}_2 \subseteq^q_{minrep} \mathcal{A}_1, \ldots, \mathcal{A}_m \subseteq^q_{minrep} \mathcal{A}_{m-1}$$

using Corollary 4.14, such that m is the least integer with the property that $|\mathcal{A}_m| \geq |\mathcal{A}_{m-1}|/2$. In other words, for all $i < m$, we have that $|\mathcal{A}_i| \leq |\mathcal{A}_{i-1}|/2$ and $|\mathcal{A}_m| \geq |\mathcal{A}_{m-1}|/2$. We output $\mathcal{A}_m$ as the q-representative family for $\mathcal{A}$. The correctness of this follows from Lemma 3.3. By Corollary 4.14,

$$\begin{aligned} |\mathcal{A}_m| &\leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log|\mathcal{A}_{m-1}| \\ &\leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log 2|\mathcal{A}_m|. \\ \text{Thus, } \frac{|\mathcal{A}_m|}{\log|\mathcal{A}_m|} &\leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)}. \end{aligned}$$

We know that for some number a and b, if $a \leq b$, then $a \log^2 a \leq b \log^2 b$. Applying this identity, we get the following:

$$\frac{|\mathcal{A}_m|}{\log |\mathcal{A}_m|} \log^2 \left(\frac{|\mathcal{A}_m|}{\log |\mathcal{A}_m|} \right) \leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)}.$$

The previous inequality implies that

$$|\mathcal{A}_m| \leq \frac{|\mathcal{A}_m|}{\log |\mathcal{A}_m|} \log^2 \left(\frac{|\mathcal{A}_m|}{\log |\mathcal{A}_m|} \right) \leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)},$$

and thus, $|\mathcal{A}_m| \leq x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)}$. By Lemma 4.14, the total running time T to compute $\mathcal{A}_m$ is

$$\begin{aligned} T &= \sum_{i=1}^{m-1} |\mathcal{A}_i| \cdot \log |\mathcal{A}_i| \cdot \log W + |\mathcal{A}_i| \cdot (1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n) \\ &= \sum_{i=1}^{m-1} \mathcal{O}\left(\frac{|\mathcal{A}|}{2^{i-1}} \cdot \log |\mathcal{A}| \cdot \log W + \frac{|\mathcal{A}|}{2^{i-1}} \cdot (1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n \right) \quad \left(\text{since } |\mathcal{A}_i| \leq \frac{|\mathcal{A}|}{2^{i-1}} \right) \\ &= \mathcal{O}\big(|\mathcal{A}| \cdot \log |\mathcal{A}| \cdot \log W + |\mathcal{A}| \cdot (1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n \big). \end{aligned}$$

This concludes the proof. □

The size of the output representative family in Theorem 4.15 is minimized when $x = \frac{p}{p+q}$. By substituting $x = \frac{p}{p+q}$ in Theorem 4.15, we get the following corollary.

COROLLARY 4.16. *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 nonnegative weight function $w : \mathcal{A} \to \mathbb{N}$ with maximum value at most W, computes in time*

$$\mathcal{O}\left(|\mathcal{A}| \cdot \log |\mathcal{A}| \cdot \log W + |\mathcal{A}| \cdot \left(\frac{p+q}{q} \right)^q \cdot 2^{o(p+q)} \cdot \log n \right)$$

a subfamily $\widehat{\mathcal{A}} \subseteq \mathcal{A}$ such that $|\widehat{\mathcal{A}}| \leq \binom{p+q}{p} \cdot 2^{o(p+q)}$ and $\widehat{\mathcal{A}} \subseteq^q_{minrep} \mathcal{A}$ ($\widehat{\mathcal{A}} \subseteq^q_{maxrep} \mathcal{A}$).

5. APPLICATIONS

In this section, we demonstrate how the efficient construction of representative families can be used to design single-exponential parameterized and exact exponential time algorithms. Our applications include the best-known deterministic algorithms for LONG DIRECTED CYCLE, MINIMUM EQUIVALENT GRAPH, k-PATH, and k-TREE.

Let $M = (E, \mathcal{I})$ be a matroid with the ground set of size n and $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a p-family of independent sets. Then, for specific matroids, we use the following notations to denote the time required to compute the following q-representative families of $\mathcal{S}$:

—$\mathcal{T}_{rm}(t, p, q)$ is the time required to compute a family $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$ of size $\binom{p+q}{q}$, when M is a linear matroid.
—$\mathcal{T}_{um}(t, p, q)$ is the time required to compute a family $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$ of size $\binom{p+q}{p} \cdot 2^{o(p+q)}$, when M is a uniform matroid and x is chosen to be $\frac{p}{p+q}$.

Let us remind that, by Theorem 1.1, when the rank of M is $p+q$, $\mathcal{T}_{rm}(t, p, q)$ is bounded by $\mathcal{O}(\binom{p+q}{p} tp^\omega + t\binom{p+q}{q}^{\omega-1})$ multiplied by the time required to perform operations over $\mathbb{F}$. By Corollary 4.16, $\mathcal{T}_{um}(t, p, q) = \mathcal{O}(t \cdot \binom{p+q}{q}^q \cdot 2^{o(p+q)} \cdot \log n)$.

5.1. Long Directed Cycle

In this section, we give our first application of algorithms based on representative families. We study the following problem:

Long Directed Cycle	**Parameter:** k
Input: An n-vertex and m-arc directed graph D and a positive integer k.	
Question: Does there exist a directed cycle of length at least k in D?	

Observe that the Long Directed Cycle problem is different from the well-known problem of finding a directed cycle of length *exactly* k. It is quite possible that the only directed cycle that has length at least k is much longer than k, and possibly even is a Hamiltonian cycle. Let D be a directed graph, k be a positive integer, and $M = (E, \mathcal{I})$ be a uniform matroid $U_{n,2k}$, where $E = V(D)$ and $\mathcal{I} = \{S \subseteq V(D) \mid |S| \leq 2k\}$. In this subsection, whenever we talk about independent sets, these are independent sets of the uniform matroid $U_{n,2k}$. For a pair of vertices $u, v \in V(D)$, we define

$$\mathcal{P}^i_{uv} = \big\{X \big| X \subseteq V(D),\ u, v \in X,\ |X| = i,\ \text{and there is a directed } uv\text{-path in } D \text{ of length } i-1 \text{ with all the vertices belonging to } X.\big\}$$

We start with a structural lemma providing the key insight to our algorithm.

Lemma 5.1. *Let D be a directed graph. Then D has a directed cycle of length at least k if and only if there exists a pair of vertices $u, v \in V(D)$ and $X \in \widehat{\mathcal{P}}^k_{uv} \subseteq^k_{rep} \mathcal{P}^k_{uv}$ such that D has a directed cycle C and in this cycle vertices of X induce a directed path (i.e., vertices of X form a consecutive segment in C).*

Proof. The reverse direction of the proof is straightforward—if cycle C contains a path of length k, the length of C is at least k. We proceed with the proof of the forward direction. Let $C^* = v_1 v_2 \cdots v_r v_1$ be a smallest directed cycle in D of length at least k. That is, $r \geq k$ and there is no directed cycle of length r' where $k \leq r' < r$. We consider two cases.

Case A: $r \leq 2k$. If $r \leq 2k$, then we take $u = v_1$ and $v = v_k$. We define paths $P = v_1 v_2 \cdots v_k$ and $Q = v_{k+1} \cdots v_r$. Because $|Q| \leq k$, by the definition of $\widehat{\mathcal{P}}^k_{uv} \subseteq^k_{rep} \mathcal{P}^k_{uv}$, there exists a directed uv-path P' such that $X = V(P') \in \widehat{\mathcal{P}}^k_{uv}$ and $X \cap Q = \emptyset$. By replacing P with P' in C^*, we obtain a directed cycle C of length at least k containing P' as a subpath.

Case B: $r \geq 2k+1$. In this case, we set $u = v_1$ and $v = v_k$ and split C^* into three paths $P = v_1 \cdots v_k$, $Q = v_{k+1} \cdots v_{2k}$, and $R = v_{2k+1} \cdots v_r$. Since $|Q| = k$ and $\widehat{\mathcal{P}}^k_{uv} \subseteq^k_{rep} \mathcal{P}^k_{uv}$, it follows that there exists an uv-path P' such that $X = V(P') \in \widehat{\mathcal{P}}^k_{uv}$ and $X \cap Q = \emptyset$. However, P' is not necessarily disjoint with R, and by replacing P with P' in C^*, we can obtain a *closed walk* C' containing P' as a subpath. See Figure 1 for an illustration.

If $X \cap R = \emptyset$, then C' is a simple cycle and we take C' as the desired C. We claim that this is the only possibility. Let us assume targeting toward a contradiction that $X \cap R \neq \emptyset$. We want to show that in this case, there is a cycle of length at least k but shorter than C^*, contradicting the choice of C^*. Let v_α be the last vertex in $X \cap R$ when we walk from v_1 to v_k along P'. Let $P'[v_\alpha, v_k]$ be the subpath of P' starting at v_α and ending at v_k. If $v_\alpha = v_{2k+1}$, we set $R' = \emptyset$. Otherwise, we put $R' = R[v_{2k+1}, v_{\alpha-1}]$ to be the subpath of R starting at v_{2k+1} and ending at $v_{\alpha-1}$. Observe that since the arc $v_{\alpha-1}v_\alpha$ is present in D (in fact, it is an arc of the cycle C^*), we have that $\overline{C} = P'[v_\alpha, v_k]QR'$ is a simple cycle in D. Clearly, $|\overline{C}| \geq |Q| \geq k$. Furthermore, since v_1 is not present in $P'[v_\alpha, v_k]$, we have that $|P'[v_\alpha, v_k]| < |P'| = |P|$. Similarly, since v_α is not present in R',

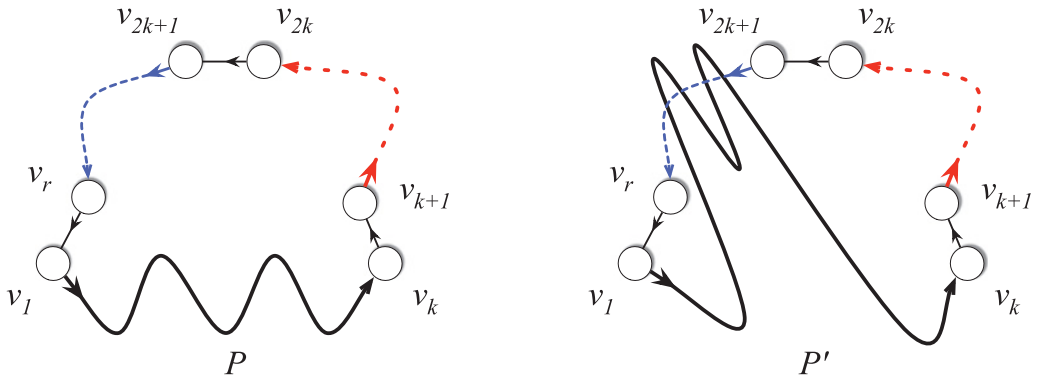

Fig. 1. Illustration to the proof of Lemma 5.1.

we have that $|R'| < |R|$. Thus, we have

$$k \leq |\overline{C}| = |P'[v_\alpha, v_k]| + |Q| + |R'| < |P| + |Q| + |R| = |C^*|.$$

This implies that $\overline{C}$ is a directed simple cycle of length at least k and strictly smaller than r. This is a contradiction. Hence, by replacing P with P' in C^*, we obtain a directed cycle C containing P' as a subpath. This concludes the proof. □

The next lemma provides an efficient computation of family $\widehat{\mathcal{P}}^k_{uv} \subseteq^k_{rep} \mathcal{P}^k_{uv}$. It is provided to give a simple exposition of the representative-families-based dynamic programming algorithm.

Lemma 5.2. *Let D be a directed/undirected graph with n vertices and m edges, $u \in V(D)$, and $M = (E, \mathcal{I})$ be a uniform matroid $U_{n,\ell}$, where $E = V(D)$ and $\mathcal{I} = \{S \subseteq V(D) \mid |S| \leq \ell\}$. Then, for every $p \leq \ell$, a collection of families $\widehat{\mathcal{P}}^p_{uv} \subseteq^{\ell-p}_{rep} \mathcal{P}^p_{uv}$, $v \in V(D) \setminus \{u\}$, of size at most*

$$\binom{\ell}{p} \cdot 2^{o(\ell)}$$

each can be found in time

$$\mathcal{O}\left(2^{o(\ell)} m \log n \max_{i \in [p]} \left\{ \binom{\ell}{i-1} \left(\frac{\ell}{\ell - i}\right)^{\ell - i} \right\}\right).$$

Furthermore, within the same running time, every set in $\widehat{\mathcal{P}}^p_{uv}$ can be ordered in a way that it corresponds to a directed (undirected) path in D.

Proof. We prove the lemma only for digraphs. The proof for undirected graphs is analogous and we only point out the differences with the proof for the directed case. We describe a dynamic programming-based algorithm. Let $V(D) = \{u, v_1, \ldots, v_{n-1}\}$ and $\mathcal{D}$ be a $(p-1) \times (n-1)$ matrix where the rows are indexed from integers in $\{2, \ldots, p\}$ and the columns are indexed from vertices in $\{v_1, \ldots, v_{n-1}\}$. The entry $\mathcal{D}[i, v]$ will store the family $\widehat{\mathcal{P}}^i_{uv} \subseteq^{\ell-i}_{rep} \mathcal{P}^i_{uv}$. We fill the entries in the matrix $\mathcal{D}$ in the increasing order of rows. For $i = 2$, $\mathcal{D}[2, v] = \{\{u, v\}\}$ if $uv \in A(D)$ (for an undirected graph we check whether u and v are adjacent). Assume that we have filled all the entries until the row i. Let

$$\mathcal{N}^{i+1}_{uv} = \bigcup_{w \in N^-(v)} \widehat{\mathcal{P}}^i_{uw} \bullet \{v\}.$$

For undirected graphs, we use the following definition:

$$\mathcal{N}_{uv}^{i+1} = \bigcup_{w \in N(v)} \widehat{\mathcal{P}}_{uw}^{i} \bullet \{v\}.$$

Claim 5.1. $\mathcal{N}_{uv}^{i+1} \subseteq_{rep}^{\ell-(i+1)} \mathcal{P}_{uv}^{i+1}$.

Proof. Let $S \in \mathcal{P}_{uv}^{i+1}$ and Y be a set of size $\ell - (i+1)$ (which is essentially an independent set of $U_{n,\ell}$) such that $S \cap Y = \emptyset$. We will show that there exists a set $S' \in \mathcal{N}_{uv}^{i+1}$ such that $S' \cap Y = \emptyset$. This will imply the desired result. Since $S \in \mathcal{P}_{uv}^{i+1}$, there exists a directed path $P = ua_1 \cdots a_{i-1}v$ in D such that $S = \{u, a_1, \ldots, a_{i-1}, v\}$ and $a_{i-1} \in N^-(v)$. The existence of path $P[u, a_{i-1}]$, the subpath of P between u and a_{i-1}, implies that $X^* = S \setminus \{v\} \in \mathcal{P}_{ua_{i-1}}^{i}$. Take $Y^* = Y \cup \{v\}$. Observe that $X^* \cap Y^* = \emptyset$ and $|Y^*| = \ell - i$. Since $\widehat{\mathcal{P}}_{ua_{i-1}}^{i} \subseteq_{rep}^{\ell-i} \mathcal{P}_{ua_{i-1}}^{i}$, there exists a set $\widehat{X}^* \in \widehat{\mathcal{P}}_{ua_{i-1}}^{i}$ such that $\widehat{X}^* \cap Y^* = \emptyset$. However, since $a_{i-1} \in N^-(v)$ and $\widehat{X}^* \cap \{v\} = \emptyset$ (as $\widehat{X}^* \cap Y^* = \emptyset$), we have $\widehat{X}^* \bullet \{v\} = \widehat{X}^* \cup \{v\}$ and $\widehat{X}^* \cup \{v\} \in \mathcal{N}_{v}^{i+1}$. Taking $S' = \widehat{X}^* \cup \{v\}$ suffices for our purpose. This completes the proof of the claim. □

We fill the entry for $\mathcal{D}[i+1, v]$ as follows. Observe that

$$\mathcal{N}_{uv}^{i+1} = \bigcup_{w \in N^-(v)} \mathcal{D}[i, w] \bullet \{v\}.$$

We already have computed the family corresponding to $\mathcal{D}[i, w]$ for $w \in N^-(v)$. By Corollary 4.16, $|\widehat{\mathcal{P}}_{uw}^{i}| \leq \binom{\ell}{i} 2^{o(\ell)}$ and thus $|\mathcal{N}_{uv}^{i+1}| \leq d^-(v)\binom{\ell}{i}2^{o(\ell)}$. Furthermore, we can compute $\mathcal{N}_{uv}^{i+1}$ in time $\mathcal{O}(d^-(v)\binom{\ell}{i}2^{o(\ell)})$. Now, using Corollary 4.16, we compute $\widehat{\mathcal{N}}_{uv}^{i+1} \subseteq_{rep}^{\ell-i-1} \mathcal{N}_{uv}^{i+1}$ in time $\mathcal{T}_{um}(t, i+1, \ell-i-1)$, where $t = d(v)\binom{\ell}{i}2^{o(\ell)}$. By Claim 5.1, we know that $\mathcal{N}_{uv}^{i+1} \subseteq_{rep}^{\ell-i-1} \mathcal{P}_{uv}^{i+1}$. Thus, Lemma 3.3 implies that $\widehat{\mathcal{N}}_{uv}^{i+1} = \widehat{\mathcal{P}}_{uv}^{i+1} \subseteq_{rep}^{\ell-i-1} \mathcal{P}_{uv}^{i+1}$. We assign this family to $\mathcal{D}[i+1, v]$. This completes the description and the correctness of the algorithm. We order the vertices of the sets in $\widehat{\mathcal{P}}_{uv}^{p}$ in the following way so that it corresponds to a directed (undirected) path in D. We keep the sets in the order in which they are built using the $\bullet$ operation. That is, we can view these sets as strings and the $\bullet$ operation as concatenation. Then, every ordered set in our family represents a path in the graph. The running time of the algorithm is bounded by

$$\begin{aligned}
&\mathcal{O}\left(\sum_{i=2}^{p}\sum_{j=1}^{n-1}\mathcal{T}_{\mathrm{um}}\left(d^-(v_j)\binom{\ell}{i-1}2^{o(\ell)}, i, \ell-i\right)\right)\\
&= \mathcal{O}\left(\sum_{i=2}^{p}\sum_{j=1}^{n-1} d^-(v_j)\binom{\ell}{i-1}\left(\frac{\ell}{\ell-i}\right)^{\ell-i} 2^{o(\ell)}\log n\right)\\
&= \mathcal{O}\left(2^{o(\ell)}\log n \sum_{i=2}^{p}\sum_{j=1}^{n-1} d^-(v_j)\binom{\ell}{i-1}\left(\frac{\ell}{\ell-i}\right)^{\ell-i}\right)\\
&= \mathcal{O}\left(2^{o(\ell)} m \log n \max_{i\in[p]}\left\{\binom{\ell}{i-1}\left(\frac{\ell}{\ell-i}\right)^{\ell-i}\right\}\right)
\end{aligned}$$

This completes the proof. □

Finally, we are ready to state the main result of this section.

THEOREM 5.3. LONG DIRECTED CYCLE *can be solved in time* $\mathcal{O}(8^{k+o(k)}mn^2)$.

PROOF. Let D be a directed graph. We solve the problem by applying the structural characterization proved in Lemma 5.1. By Lemma 5.1, D has a directed cycle of length at least k if and only if there exists a pair of vertices $u, v \in V(D)$ and a path P' with $V(P') \in \widehat{\mathcal{P}}^k_{uv} \subseteq^k_{rep} \mathcal{P}^k_{uv}$ such that D has a directed cycle C containing P' as a subpath.

We first compute $\widehat{\mathcal{P}}^k_{uv} \subseteq^k_{rep} \mathcal{P}^k_{uv}$ for all $u, v \in V(D)$. For that we apply Lemma 5.2 for each vertex $u \in V(D)$ with $\ell = 2k$ and $p = k$. Thus, we can compute $\widehat{\mathcal{P}}^k_{uv} \subseteq^k_{rep} \mathcal{P}^k_{uv}$ for all $u, v \in V(D)$ in time $\mathcal{O}(2^{o(k)}m \log n \max_{i\in[k]}\{\binom{2k}{i-1}(\frac{2k}{2k-i})^{2k-i}\})$. Both the functions $\binom{2k}{i-1}$ and $(\frac{2k}{2k-i})^{2k-i}$ on i in the domain $[k]$ are maximized when $i = k$. Thus, the mentioned running time is upper bounded by $\mathcal{O}(8^{k+o(k)}mn \log n)$. Moreover, for every $X \in \widehat{\mathcal{P}}^k_{uv}$, we also compute a directed uv-path P_X using vertices of X. Let

$$\mathcal{Q} = \bigcup_{u,v\in V(D)} \widehat{\mathcal{P}}^k_{uv}.$$

Now, for every set $X \in \mathcal{Q}$ and the corresponding uv-path P_X with an endpoint, we check if there is a uv-path in D avoiding all vertices of X but u and v. This check can be done by a standard graph traversal algorithm like BFS/DFS in time $\mathcal{O}(m + n)$. If we succeed in finding a path for at least one $X \in \mathcal{Q}$, we answer YES and return the corresponding directed cycle obtained by merging P_X and another path. Otherwise, if we did not succeed to find such a path for any of the sets $X \in \mathcal{Q}$, this means that there is no directed cycle of length at least k in D. The correctness of the algorithm follows from Lemma 5.1. By Corollary 4.16, the size of $\mathcal{Q}$ is upper bounded by $n^2\binom{2k}{k}2^{o(k)} \leq n^2 4^{k+o(k)}$. Thus, the overall running time of the algorithm is upper bounded by

$$\mathcal{O}\big(8^{k+o(k)}mn \log n + 4^{k+o(k)}(n^2m + n^3)\big).$$

This concludes the proof. □

5.2. Faster Long Directed Cycle

In this subsection, we design a faster algorithm for LONG DIRECTED CYCLE. In Section 5.1, we saw an algorithm for LONG DIRECTED CYCLE where the running time mainly depends on the computation of representative families $\widehat{\mathcal{P}}^p_{uv} \subseteq^q_{rep} \mathcal{P}^p_{uv}$ for $2 \leq p \leq k$ and $q = 2k - p$. We used Theorem 4.12 with $x = \frac{p}{p+q}$ (i.e., Corollary 4.16) to compute representative families. The choice $x = \frac{p}{p+q}$ minimizes the size of the representative family. But in fact, we can choose x that minimizes the running time instead.

Now we find out the choice of x that minimizes the computation of $\widehat{\mathcal{P}}^p_{uv} \subseteq^q_{rep} \mathcal{P}^p_{uv}$ for $2 \leq p \leq k$ and $q = 2k - p$. Let $s_{p,q}$ denote the size of $\widehat{\mathcal{P}}^p_{uv}$. We know that the computation of $\widehat{\mathcal{P}}^p_{uv} \subseteq^q_{rep} \mathcal{N}^p_{uv} \subseteq^q_{rep} \mathcal{P}^p_{uv}$ depends on $|\mathcal{N}^p_{uv}|$, which depends on the size of the representative families $\widehat{\mathcal{P}}^{p-1}_{uw}$. That is, $|\mathcal{N}^p_{uv}| \leq s_{p-1,q+1} \cdot n$. Thus, the values of $s_{p-1,q+1}$ and $s_{p,q}$ are "almost equal" and we denote it by $s_{p-1,q+1} \approx s_{p,q}$. By Theorem 4.15, the running time to compute $\widehat{\mathcal{P}}^p_{uv} \subseteq^q_{rep} \mathcal{N}^p_{uv} \subseteq^q_{rep} \mathcal{P}^p_{uv}$ is

$$\begin{aligned}
&\mathcal{O}\left(|\mathcal{N}^p_{uv}| \cdot (1-x)^{-q} \cdot 2^{o(p+q)} \cdot \log n\right) \\
&= \mathcal{O}\left(s_{p,q} \cdot (1-x)^{-q} \cdot 2^{o(p+q)} \cdot n \log n\right) \\
&= \mathcal{O}\left(x^{-p} \cdot (1-x)^{-2q} \cdot 2^{o(p+q)} \cdot n \log n\right).
\end{aligned}$$

To minimize the previous running time, it is enough to minimize the function $f(x) = x^{-p} \cdot (1-x)^{-2q}$. Using methods from calculus, we know that the value x^* of x for which $f'(x^*) = 0$ corresponds to a minimum value of the function $f(x)$ if $f''(x^*) > 0$. The derivative of $f(x)$ is $f'(x) = -px^{-p-1}(1-x)^{-2q} + 2q \cdot x^{-p}(1-x)^{-2q-1}$. Now consider the value of x for which $f'(x) = 0$:

$$\begin{aligned} -px^{-p-1}(1-x)^{-2q} + 2q \cdot x^{-p}(1-x)^{-2q-1} &= 0 \\ -p(1-x) + 2q \cdot x &= 0 \\ x &= \frac{p}{p+2q}. \end{aligned}$$

Set $x^* = \frac{p}{p+2q}$. To prove $f(x)$ is minimized at x^*, it is enough to show that $f''(x^*) > 0$:

$$\begin{aligned} f'(x) &= -px^{-p-1}(1-x)^{-2q} + 2q \cdot x^{-p}(1-x)^{-2q-1} \\ &= x^{-p}(1-x)^{-2q}(-p \cdot x^{-1} + 2q \cdot (1-x)^{-1}) \\ &= f(x) \cdot (-p \cdot x^{-1} + 2q \cdot (1-x)^{-1}) \\ f''(x) &= f(x) \cdot (p \cdot x^{-2} + 2q \cdot (1-x)^{-2}) + f'(x) \cdot (-p \cdot x^{-1} + 2q \cdot (1-x)^{-1}) \\ f''(x^*) &= f(x^*) \cdot (p \cdot (x^*)^{-2} + 2q \cdot (1-(x^*))^{-2}) > 0. \end{aligned}$$

Hence, the runtime to compute $\widehat{\mathcal{P}}^p_{uv} \subseteq^q_{rep} \mathcal{P}^p_{uv}$ is minimized when $x = \frac{p}{p+2q}$.

LEMMA 5.4. *Let D be a directed graph with n vertices and m edges, $u \in V(D)$, and $M = (E, \mathcal{I})$ be a uniform matroid $U_{n,\ell}$, where $E = V(D)$ and $\mathcal{I} = \{S \subseteq V(D) \mid |S| \leq \ell\}$. Then, for every $v \in V(D) \setminus \{u\}$ and integer $2 \leq p \leq \ell$, there is an algorithm that computes a family $\widehat{\mathcal{P}}^p_{uv} \subseteq^{\ell-p}_{rep} \mathcal{P}^p_{uv}$ of size $(\frac{2\ell-p}{p})^p(\frac{2\ell-p}{2\ell-2p})^{\ell-p} \cdot 2^{o(\ell)}$ in time $\mathcal{O}(2^{o(\ell)} \cdot m \log n \cdot \max_{i\in[p]}\{(\frac{2\ell-i}{i})^i(\frac{2\ell-i}{2\ell-2i})^{2\ell-2i}\})$.*

PROOF. The proof is the same as the proof of Lemma 5.2, except for the choice of x while applying Theorem 4.12 (instead of Corollary 4.16). As in the proof of Lemma 5.2, we have a dynamic programming table $\mathcal{D}$ where the rows are indexed from integers in $\{2, \ldots, p\}$ and the columns are indexed from vertices in $\{v_1, \ldots, v_{n-1}\}$. The entry $\mathcal{D}[i, v]$ will store the family $\widehat{\mathcal{P}}^i_{uv} \subseteq^{\ell-i}_{rep} \mathcal{P}^i_{uv}$. We fill the entries in the matrix $\mathcal{D}$ in the increasing order of rows. For $i = 2$, $\mathcal{D}[2, v] = \{\{u, v\}\}$ if $uv \in A(D)$. Assume that we have filled all the entries until the row i. Let

$$\mathcal{N}^{i+1}_{uv} = \bigcup_{w \in N^-(v)} \widehat{\mathcal{P}}^i_{uw} \bullet \{v\}.$$

Due to Claim 5.1, we have that $\mathcal{N}^{i+1}_{uv} \subseteq^{\ell-(i+1)}_{rep} \mathcal{P}^{i+1}_{uv}$. Lemma 3.3 implies that $\widehat{\mathcal{N}}^{i+1}_{uv} = \widehat{\mathcal{P}}^{i+1}_{uv} \subseteq^{\ell-i-1}_{rep} \mathcal{P}^{i+1}_{uv}$. We assign this family to $\mathcal{D}[i+1, v]$.

Now we explain the computation of $\widehat{\mathcal{N}}^{i+1}_{uv} = \widehat{\mathcal{P}}^{i+1}_{uv}$. For any j, to compute $\widehat{\mathcal{N}}^j_{uv} = \widehat{\mathcal{P}}^j_{uv}$, we apply Theorem 4.15 with the value x_j for x, where

$$x_j = \frac{j}{j + 2(\ell - j)} = \frac{j}{2\ell - j}.$$

Let $s_{j,\ell-j}$ be the size of the representative family $\widehat{\mathcal{N}}^j_{uv} = \widehat{\mathcal{P}}^j_{uv}$ when we apply Theorem 4.15 with the value x_j. That is, $s_{j,\ell-j} = (x_j)^{-j}(1-x_j)^{\ell-j} \cdot 2^{o(\ell)}$. Assume that we have computed $\widehat{\mathcal{P}}^j_{uw}$ of size $s_{j,\ell-j}$ and stored it in $\mathcal{D}[j, w]$ for all $j \leq i$ and $w \in \{v_1, \ldots, v_{n-1}\}$. Now consider the computation of $\widehat{\mathcal{N}}^{i+1}_{uv} = \widehat{\mathcal{P}}^{i+1}_{uv}$. We apply Theorem 4.15 with value x_{i+1} for x to compute

$\widehat{\mathcal{N}}_{uv}^{i+1} \subseteq_{rep}^{\ell-(i+1)} \mathcal{N}_{uv}^{i+1}$. Since $\mathcal{N}_{uv}^{i+1} = \bigcup_{w \in N^-(v)} \widehat{\mathcal{P}}_{uw}^{i} \bullet \{v\}$, we have that

$$\begin{aligned} |\mathcal{N}_{uv}^{i+1}| &\leq s_{i,\ell-i} \cdot d^-(v) \\ &\leq (x_i)^{-i}(1-x_i)^{\ell-i} \cdot 2^{o(\ell)} d^-(v). \end{aligned}$$

By Theorem 4.15, the running time to compute $\widehat{\mathcal{N}}_{uv}^{i+1}$ is

$$s_{i,\ell-i} \cdot (1-x_{i+1})^{\ell-(i+1)} \cdot 2^{o(\ell)} \cdot d^-(v) \cdot \log n. \tag{6}$$

To analyze the running time further, we need the following claim.

CLAIM 5.2. *For any $3 < i < p$, $s_{i,\ell-i} \leq e^2 \cdot (i+1) \cdot s_{i+1,\ell-i-1}$.*

PROOF. By applying the definition of s_i and x_{i+1}, we get the following inequality:

$$\begin{aligned} \frac{s_{i,\ell-i}}{s_{i+1,\ell-i-1}} &= \frac{x_i^{-i}(1-x_i)^{-\ell+i}}{x_{i+1}^{-(i+1)}(1-x_{i+1})^{-\ell+(i+1)}} \\ &= \left(\frac{2\ell-i}{i}\right)^{i} \left(\frac{2\ell-i}{2\ell-2i}\right)^{\ell-i} \left(\frac{i+1}{2\ell-(i+1)}\right)^{i+1} \left(\frac{2\ell-2(i+1)}{2\ell-(i+1)}\right)^{\ell-(i+1)} \\ &= \left(\frac{2\ell-i}{2\ell-(i+1)}\right)^{\ell} \cdot \frac{(i+1)^{i+1}}{i^i} \cdot \frac{(2\ell-2(i+1))^{\ell-(i+1)}}{(2\ell-2i)^{\ell-i}} \\ &\leq \left(1+\frac{1}{2\ell-(i+1)}\right)^{2\ell-(i+1)} \cdot (i+1) \cdot \left(1+\frac{1}{i}\right)^{i} \\ &\leq e^2 \cdot (i+1). \end{aligned}$$

In the last transition, we used that $(1+1/x)^x < e$ for every $x > 0$. □

From Equation (6) and Claim 5.2, we have that the running time for computing $\widehat{\mathcal{P}}_{uv}^{p}$ is bounded by

$$\begin{aligned} &\mathcal{O}\left(\sum_{i=2}^{p}\sum_{j=1}^{n-1} s_{i,\ell-i} \cdot d^-(v_j) \cdot (1-x_i)^{-\ell+i} \cdot 2^{o(\ell)} \cdot \log n\right) \\ &= \mathcal{O}\left(2^{o(\ell)} \cdot m \log n \cdot \max_{i\in[p]}\left\{\left(\frac{2\ell-i}{i}\right)^{i}\left(\frac{2\ell-i}{2\ell-2i}\right)^{2\ell-2i}\right\}\right). \end{aligned}$$

The size of the family $\widehat{\mathcal{P}}_{uv}^{p} \subseteq_{rep}^{\ell-p} \mathcal{N}_{uv}^{p} \subseteq_{rep}^{\ell-p} \mathcal{P}_{uv}^{p}$ is

$$s_{p,\ell-p} = (x_p)^{-p}(1-x_p)^{-\ell+p} \cdot 2^{o(\ell)} = \left(\frac{2\ell-p}{p}\right)^{p}\left(\frac{2\ell-p}{2\ell-2p}\right)^{\ell-p} \cdot 2^{o(\ell)}.$$

This completes the proof.

We now have a faster algorithm to compute the representative family $\widehat{\mathcal{P}}_{uv}^{k} \subseteq_{rep}^{k} \mathcal{P}_{uv}^{p}$. Using Lemma 5.4, we can compute $\widehat{\mathcal{P}}_{uv}^{k}$ for all $v \in V(D) \setminus \{u\}$ in time

$$\mathcal{O}\left(2^{o(k)} \cdot m \log n \cdot \max_{i\in[p]}\left\{\left(\frac{4k-i}{i}\right)^{i}\left(\frac{4k-i}{4k-2i}\right)^{4k-2i}\right\}\right).$$

Simple calculus shows that the maximum is attained for $i = k$. Hence, the running time to compute $\widehat{\mathcal{P}}_{uv}^{k}$ for all $u, v \in V(D)$ is upper bounded by $\mathcal{O}(6.75^{k+o(k)} nm \log n)$. This yields an improved bound for the running time of our algorithm for LONG DIRECTED CYCLE.

We apply Lemma 5.4 for each $u \in V(D)$ with $\ell = 2k$ and $p = k$. Thus, we can compute $\widehat{\mathcal{P}}^k_{uv} \subseteq^k_{rep} \mathcal{P}^p_{uv}$ for all $u, v \in V(D)$ in time $\mathcal{O}(6.75^{k+o(k)}nm \log n)$. The size of the family $\widehat{\mathcal{P}}^k_{uv}$ for any $u, v \in V(D)$ is upper bounded by $\mathcal{O}(4.5^{k+o(k)})$. Thus, if we now loop over every set in the representative families and run a breadth-first search, just as in the proof of Theorem 5.3, this will take at most $\mathcal{O}(6.75^{k+o(k)}nm \log n + 4.5^{k+o(k)}(n^3 + n^2m))$ time. Hence, we arrive at the following theorem.

THEOREM 5.5. *There is a $\mathcal{O}(6.75^{k+o(k)}mn^2)$ time algorithm for* LONG DIRECTED CYCLE.

5.3. Minimum Equivalent Graph

For a given digraph D, a subdigraph D' of D is said to be an *equivalent* subdigraph of D if for any pair of vertices $u, v \in V(D)$, if there is a directed path in D from u to v, then there is also a directed path from u to v in D'. That is, reachability of vertices in D and D' is the same. In this section, we study a problem where given a digraph D, the objective is to find an equivalent subdigraph D' of D with as few arcs as possible. Equivalently, the objective is to remove the maximum number of arcs from a digraph D without affecting its reachability. More precisely, the problem we study is as follows:

MINIMUM EQUIVALENT GRAPH (MEG)
Input: A directed graph D.
Task: Find an equivalent subdigraph of D with the minimum number of arcs.

The following proposition is due to Moyles and Thompson [1969] (see also Bang-Jensen and Gutin [2009, Section 2.3]), who reduce the problem of finding a minimum equivalent subdigraph of an arbitrary D to a strong digraph.

PROPOSITION 5.6. *Let D be a digraph on n vertices with strongly connected components $C_1, \ldots, C_r$. Given a minimum equivalent subdigraph C'_i for each C_i, $i \in [r]$, one can obtain a minimum equivalent subdigraph D' of D containing each C'_i in $\mathcal{O}(n^\omega)$ time.*

Observe that for a strong digraph D, any equivalent subdigraph is also strong. By Proposition 5.6, MEG reduces to the following problem:

MINIMUM STRONGLY CONNECTED SPANNING SUBGRAPH (MINIMUM SCSS)
Input: A strongly connected directed graph D.
Task: Find a strong spanning subdigraph of D with the minimum number of arcs.

There seems to be no established agreement in the literature on what to call these problems. MEG sometimes is also referred as MINIMUM EQUIVALENT DIGRAPH and MINIMUM EQUIVALENT SUBDIGRAPH, while MINIMUM SCSS is also called MINIMUM SPANNING STRONG SUBDIGRAPH (MSSS).

A digraph T is an *out-tree* (an *in-tree*) if T is an oriented tree with just one vertex s of in-degree zero (out-degree zero). The vertex s is the root of T. If an out-tree (in-tree) T is a spanning subdigraph of D, T is called an *out-branching* (an *in-branching*). We use the notation B^+_s (B^-_s) to denote an out-branching (in-branching) rooted at s of the digraph. A digraph F is an *out-forest* (an *in-forest*) if it is a disjoint union of out-trees (in-trees).

It is known that a digraph is strong if and only if it contain an out-branching and an in-branching rooted at some vertex $v \in V(D)$ [Bang-Jensen and Gutin 2009, Proposition 12.1.1].

PROPOSITION 5.7. *Let D be a strong digraph on n vertices, let v be an arbitrary vertex of $V(D)$, and let $\ell \leq n - 2$ be a natural number. Then there exists a strong spanning*

subdigraph of D with at most $2n-2-\ell$ arcs if and only if D contains an in-branching B_v^- and an out-branching B_v^+ with root v so that $|A(B_v^+) \cap A(B_v^-)| \geq \ell$ (i.e., they have at least ℓ common arcs).

Proposition 5.7 implies that the MINIMUM SCSS problem is equivalent to finding, for an arbitrary vertex $v \in V(D)$, an out-branching B_v^+ and an in-branching B_v^- that maximizes $|A(B_v^+) \cap A(B_v^-)|$. For our exact algorithm for MINIMUM SCSS, we implement this equivalent version using representative families.

Let D be a strong digraph and $s \in V(D)$ be a fixed vertex. For $v \in V(D)$, we use $\mathsf{In}(v)$ and $\mathsf{Out}(v)$ to denote the sets of incoming and outgoing arcs incident with v in D, respectively. By D_s^- we denote the digraph obtained from D by deleting the arcs in $\mathsf{Out}(s)$. Similarly, by D_s^+ we denote the digraph obtained from D by deleting the arcs in $\mathsf{In}(s)$.

We take two copies E_1, E_2 of $A(D)$ (i.e., $E_i = \{e_i \mid e \in A(D)\}$), a copy E_3 of $A(D_s^+)$, and a copy E_4 of $A(D_s^-)$ and construct four matroids as follows. Let $U(D)$ denote the underlying undirected graph of D. The first two matroids $M_1 = (E_1, \mathcal{I}_1)$, $M_2 = (E_2, \mathcal{I}_2)$ are the graphic matroids on $U(D)$. Observe that

$$A(D_s^+) = \biguplus_{v \in V(D_s^+) \setminus \{s\}} \mathsf{In}(v) \text{ and } A(D_s^-) = \biguplus_{v \in V(D_s^-) \setminus \{s\}} \mathsf{Out}(v).$$

Thus, the arcs of D_s^+ can be partitioned into sets of in-arcs and similarly the arcs of D_s^- into sets of out-arcs. The other two matroids are the following partition matroids $M_3 = (E_3, \mathcal{I}_3)$, $M_4 = (E_4, \mathcal{I}_4)$, where

$$\mathcal{I}_3 = \{I \mid I \subseteq A(D_s^+), \text{ for every } v \in V(D_s^+) = V(D), |I \cap \mathsf{In}(v)| \leq 1\},$$

and

$$\mathcal{I}_4 = \{I \mid I \subseteq A(D_s^-), \text{ for every } v \in V(D_s^-) = V(D), |I \cap \mathsf{Out}(v)| \leq 1\}.$$

We define the matroid $M = (E, \mathcal{I})$ as the direct sum $M = M_1 \oplus M_2 \oplus M_3 \oplus M_4$. Since each M_i is a representable matroid over the same field (by Propositions 2.4 and 2.5), we have that M is also representable (Proposition 2.3). The reason we say that M_i is representable over the same field $\mathbb{F}$ is that the graphic matroid is representable over any field and the partition matroids defined here are representable over a finite field of size $n^{\mathcal{O}(1)}$. So if we take $\mathbb{F}$ as a finite field of size $n^{\mathcal{O}(1)}$, then M is representable over $\mathbb{F}$. The rank of this matroid is $4n-4$.

Let us note that for each arc $e \in A(D)$ that is not incident with s, we have four elements in the matroid M, corresponding to the copies of e in M_i, $i \in \{1, \ldots, 4\}$. We denote these elements by e_i, $i \in \{1, \ldots, 4\}$. For every edge $e \in A(D)$ incident with s, we have three corresponding elements. We denote them by e_1, e_2, e_3 or e_1, e_2, e_4, depending on the case when e is an in- or out-arc for s.

For $i \in \{1, \ldots, n-1\}$, we define

$$\mathcal{B}^{4i} = \left\{W \in \mathcal{I} \mid |W| = 4i, \forall e \in A(D) \text{ either } W \cap \{e_1, e_2, e_3, e_4\} = \emptyset \text{ or } \{e_1, e_2, e_3, e_4\} \subseteq W\right\}.$$

For $W \in \mathcal{I}$, by A_W we denote the set of arcs $e \in A(D)$ such that $\{e_1, e_2, e_3, e_4\} \cap W \neq \emptyset$. Now we are ready to state the lemma that relates representative families and the MINIMUM SCSS problem.

LEMMA 5.8. *Let D be a strong digraph on n vertices and $\ell \leq n-2$ be a natural number. Then there exists a strong spanning subdigraph D' of D with at most $2n-2-\ell$ arcs if and only if there exists a set $\widehat{\mathcal{F}} \in \widehat{\mathcal{B}}^{4\ell} \subseteq_{rep}^{n'-4\ell} \mathcal{B}^{4\ell}$ such that D has a strong spanning subdigraph $\bar{D}$ with $A_{\widehat{\mathcal{F}}} \subseteq A(\bar{D})$ and $|A(\bar{D})| \leq 2n-2-\ell$. Here, $n' = 4n-4$.*

PROOF. We only show the forward direction of the proof; the reverse direction is straightforward. Let D' be a strong spanning subdigraph of D with at most $2n-2-\ell$ arcs. Thus, by Proposition 5.7, we have that for any vertex $v \in V(D')$, there exists an out-branching B_v^+ and an in-branching B_v^- in D' such that $|A(B_v^+) \cap A(B_v^-)| \geq \ell$. Observe that the arcs in $A(B_v^+) \cap A(B_v^-)$ form an out-forest (in-forest). Let F' be an arbitrary subset of $A(B_v^+) \cap A(B_v^-)$ containing exactly ℓ arcs. Take $X = A(B_v^+) \setminus F'$ and $Y = A(B_v^-) \setminus F'$. Observe that X and Y need not be disjoint. Clearly, $|X| = |Y| = n-1-\ell$.

In matroid M, one can associate with D' an independent set $I_{D'}$ of size $4n-4$ as follows:

$$I_{D'} = \bigcup_{e \in F'} \{e_1, e_2, e_3, e_4\} \bigcup_{e \in X} \{e_1, e_3\} \bigcup_{e \in Y} \{e_2, e_4\}.$$

By our construction, we have that $I_{D'}$ is an independent set in $\mathcal{I}$ and $|I_{D'}| = 4\ell + 4(n-1-\ell) = n'$. Let $F = \bigcup_{e \in F'} \{e_1, e_2, e_3, e_4\}$, $\bar{X} = \bigcup_{e \in X} \{e_1, e_3\}$, and $\bar{Y} = \bigcup_{e \in Y} \{e_2, e_4\}$. Then notice that $F \in \mathcal{B}^{4\ell}$ and $F \subset I_{D'}$. This implies that there exists a set $\widehat{F} \in \widehat{\mathcal{B}}^{4\ell} \subseteq_{rep}^{n'-4\ell} \mathcal{B}^{4\ell}$ such that $I_{\bar{D}} = \widehat{F} \cup \bar{X} \cup \bar{Y} \in \mathcal{I}$. We show that D has a strong spanning subdigraph $\bar{D}$ with $A_{\widehat{F}} \subseteq A(\bar{D})$. Let $\bar{D}$ be the digraph with the vertex set $V(D)$ and the arc set $A(\bar{D}) = X \cup Y \cup A_{\widehat{F}}$. Notice that $|A(\bar{D})| = |A(D')| \leq 2n-2-\ell$. Consider the following four sets:

(1) Let $W_1 = \{e_1 \mid e \in X \cup A_{\widehat{F}}\}$; then we have that $W_1 \subseteq I_{\bar{D}}$ and thus $W_1 \in \mathcal{I}_1$. This together with the fact that $|W_1| = n-1$ implies that $X \cup A_{\widehat{F}}$ forms a spanning tree in $U(D)$.
(2) Let $W_2 = \{e_2 \mid e \in Y \cup A_{\widehat{F}}\}$. Similar to the first case, then $Y \cup A_{\widehat{F}}$ forms a spanning tree in $U(D)$.
(3) Let $W_3 = \{e_3 \mid e \in X \cup A_{\widehat{F}}\}$; then we have that $W_3 \subseteq I_{\bar{D}}$ and thus $W_3 \in \mathcal{I}_3$. This together with the fact that $|W_1| = |W_3| = n-1$ and that $X \cup A_{\widehat{F}}$ is a spanning tree in $U(D)$ implies that $X \cup A_{\widehat{F}}$ forms an out-branching rooted at s in D_s^+.
(4) Let $W_4 = \{e_3 \mid e \in Y \cup A_{\widehat{F}}\}$. Similar to the previous case, then $Y \cup A_{\widehat{F}}$ forms an in-branching rooted at s in D_s^-.

We have shown that $\bar{D}$ contains $A_{\widehat{F}}$ and has an out-branching and in-branching rooted at s. Also, $|A(\bar{D})| \leq 2n-2-\ell$. This implies that $\bar{D}$ is the desired strong spanning subdigraph of D containing a set from $\widehat{\mathcal{B}}^{4\ell}$. This concludes the proof of the lemma. □

LEMMA 5.9. *Let D be a strong digraph on n vertices and $\ell \leq n-2$ be a natural number. Then, in time $\mathcal{O}(\max_{i \in [\ell]} \binom{n'}{4i}^{\omega} mn^2 \log n)$, we can compute $\widehat{\mathcal{B}}^{4\ell} \subseteq_{rep}^{n'-4\ell} \mathcal{B}^{4\ell}$ of size $\binom{n'}{4\ell}$. Here, $n' = 4n-4$.*

PROOF. We describe a dynamic programming-based algorithm. Let $\mathcal{D}$ be an array of size ℓ. The entry $\mathcal{D}[i]$ will store the family $\widehat{\mathcal{B}}^{4i} \subseteq_{rep}^{n'-4i} \mathcal{B}^{4\ell}$. We fill the entries in the array $\mathcal{D}$ in the increasing order of its index, that is, from $0, \ldots, \ell$. For the base case, define $\widehat{\mathcal{B}}^0 = \{\emptyset\}$ and let $W = \{\{e_1, e_2, e_3, e_4\} \mid e \in A(D)\}$. Given that $\mathcal{D}[i]$ is filled for all $i' \leq i$, we fill $\mathcal{D}[i+1]$ as follows. Define $\mathcal{N}^{4(i+1)} = (\widehat{\mathcal{B}}^{4i} \bullet W) \cap \mathcal{I}$.

CLAIM 5.3. *For all $0 \leq i \leq \ell - 1$, $\mathcal{N}^{4(i+1)} \subseteq_{rep}^{n'-4(i+1)} \mathcal{B}^{4(i+1)}$.*

PROOF. Let $S \in \mathcal{B}^{4(i+1)}$ and Y be a set of size $n' - 4(i+1)$ such that $S \cap Y = \emptyset$ and $S \cup Y \in \mathcal{I}$. We will show that there exists a set $\hat{S} \in \mathcal{N}^{4(i+1)}$ such that $\hat{S} \cap Y = \emptyset$ and $\hat{S} \cup Y \in \mathcal{I}$. This will imply the desired result.

Let $e \in A(D)$ such that $\{e_1, e_2, e_3, e_4\} \subseteq S$. Define $S^* = S \setminus \{e_1, e_2, e_3, e_4\}$ and $Y^* = Y \cup \{e_1, e_2, e_3, e_4\}$. Since $S \cup Y \in \mathcal{I}$, we have that $S^* \in \mathcal{I}$ and $Y^* \in \mathcal{I}$. Observe that $S^* \in \mathcal{B}^{4i}$, $S^* \cup Y^* \in \mathcal{I}$ and the size of Y^* is $n' - 4i$. This implies that there exists $\widehat{S}^*$ in

$\widehat{\mathcal{B}}^{4i} \subseteq_{rep}^{n'-4i} \mathcal{B}^{4\ell}$ such that $\widehat{S}^* \cup Y^* \in \mathcal{I}$. Thus, $\widehat{S}^* \cup \{e_1, e_2, e_3, e_4\} \in \mathcal{I}$ and also in $\widehat{\mathcal{B}}^{4i} \bullet W$ and thus in $\mathcal{N}^{4(i+1)}$. Taking $\widehat{S} = \widehat{S}^* \cup \{e_1, e_2, e_3, e_4\}$ suffices for our purpose. This completes the proof of the claim. □

We fill the entry for $\mathcal{D}[i+1]$ as follows. Observe that $\mathcal{N}_{uv}^{4(i+1)} = (\mathcal{D}[i] \bullet W) \cap \mathcal{I}$. We already have computed the family corresponding to $\mathcal{D}[i]$. By Theorem 1.1, $|\widehat{\mathcal{B}}^{4i}| \leq \binom{n'}{4i}$ and thus $|\mathcal{N}^{4(i+1)}| \leq 4m\binom{n'}{4i}$. Furthermore, we can compute $\mathcal{N}^{4(i+1)}$ in time $\mathcal{O}(mn\binom{n'}{4i})$. Now, using Theorem 1.1, we can compute $\widehat{\mathcal{N}}^{4(i+1)} \subseteq_{rep}^{n'-4(i+1)} \mathcal{N}^{4(i+1)}$ in time $\mathcal{T}_{rm}(t, 4i+4, n'-4(i+1))$, where $t = 4m\binom{n'}{4i}$.

By Claim 5.3, we know that $\mathcal{N}^{4(i+1)} \subseteq_{rep}^{n'-4(i+1)} \mathcal{B}^{4(i+1)}$. Thus, Lemma 3.3 implies that $\widehat{\mathcal{N}}^{4(i+1)} = \widehat{\mathcal{B}}^{4(i+1)} \subseteq_{rep}^{n'-4(i+1)} \mathcal{B}^{4(i+1)}$. We assign this family to $\mathcal{D}[i+1]$. This completes the description and the correctness of the dynamic programming. The field size for uniform matroids is upper bounded by $n^{\mathcal{O}(1)}$ and thus we can perform all the field operations in time $\mathcal{O}(\log^2 n)$. Thus, the running time of this algorithm is upper bounded by

$$\mathcal{O}\left(\sum_{i=1}^{\ell} \mathcal{T}_{rm}\left(4m\binom{n'}{4(i-1)}, 4i, n'-4i\right)\right) = \mathcal{O}\left(\max_{i\in[\ell]}\binom{n'}{4i}^{\omega} m\log^2 n\right).$$

This completes the proof.

Lemma 5.10. Minimum SCSS *can be solved in time* $\mathcal{O}(2^{4\omega n}mn)$.

Proof. Let us fix $n' = 4n-4$. Proposition 5.7 implies that the Minimum SCSS problem is equivalent to finding, for an arbitrary vertex $s \in V(D)$, an out-branching B_v^+ and an in-branching B_v^- that maximizes $|A(B_v^+) \cap A(B_v^-)|$. We guess the value of $|A(B_v^+) \cap A(B_v^-)|$ and let this be ℓ. By Lemma 5.8, there exists a strong spanning subdigraph D' of D with at most $2n-2-\ell$ arcs if and only if there exists a set $\widehat{F} \in \widehat{\mathcal{B}}^{4\ell} \subseteq_{rep}^{n'-4\ell} \mathcal{B}^{4\ell}$ such that D has a strong spanning subdigraph $\bar{D}$ with $A_{\widehat{F}} \subseteq A(\bar{D})$. Recall that for $X \in \mathcal{I}$, by A_X, we denote the set of arcs $e \in A(D)$ such that $\{e_1, e_2, e_3, e_4\} \cap X \neq \emptyset$. Now, using Lemma 5.9, we compute $\widehat{\mathcal{B}}^{4\ell} \subseteq_{rep}^{n'-4\ell} \mathcal{B}^{4\ell}$ in time $\mathcal{O}(\max_{i\in[\ell]}\binom{n'}{4i}^{\omega} m\log^2 n)$.

For every $\widehat{F} \in \widehat{\mathcal{B}}^{4\ell}$, we test whether $A_{\widehat{F}}$ can be extended to an out-branching in D_s^+ and to an in-branching in D_s^-. We can do it in $\mathcal{O}(n(n+m))$-time by putting weights 0 to the arcs of $A_{\widehat{F}}$ and weights 1 to all remaining arcs and then by running the classic algorithm of Edmonds [1967]. Since $\ell \leq n-2$, the running time of this algorithm is upper bounded by $\mathcal{O}(2^{4\omega n}mn)$. This concludes the proof. □

Finally, we are ready to prove the main result of this section.

Theorem 5.11. Minimum Equivalent Graph *can be solved in time* $\mathcal{O}(2^{4\omega n}mn)$.

Proof. Given an arbitrary digraph D, we first find its strongly connected components $C_1, \ldots, C_s$. Now, on each C_i, we apply Lemma 5.10 and obtain a minimum equivalent subdigraph C_i'. After this we apply Proposition 5.6 and obtain a minimum equivalent subdigraph of D. Since all the steps except Lemma 5.10 take polynomial time, we get the desired running time. This completes the proof. □

A weighted variant of Minimum Equivalent Graph has also been studied in the literature. More precisely, the problem is defined as follows:

MINIMUM WEIGHT EQUIVALENT GRAPH (MWEG)
Input: A directed graph D and a weight function $w : A(D) \to \mathbb{N}$.
Task: Find a minimum weight equivalent subdigraph of D.

MWEG can be solved along the same lines as MEG, but to do this, we need to use the notion of the min q-representative family and use Theorem 3.7 instead of Theorem 1.1. We also need a proposition like Proposition 5.6 for the weighted case. If D is an arc weighted graph, then one can get a proposition similar to Proposition 5.6 with a $\log W$ multiplicative factor in the running time, where W is the largest arc weight in D. These changes give us the following theorem.

THEOREM 5.12. MINIMUM WEIGHT EQUIVALENT GRAPH *can be solved in time* $\mathcal{O}(2^{4\omega n}mn \log W)$. *Here,* W *is the maximum value assigned by the weight function* $w : A(D) \to \mathbb{N}$.

5.4. Dynamic Programming Over Graphs of Bounded Treewidth

In this section, we discuss deterministic algorithms for "connectivity problems" such as HAMILTONIAN PATH, STEINER TREE, and FEEDBACK VERTEX SET parameterized by the treewidth of the input graph. The algorithms are based on Theorem 1.1 and use graphic matroids to take care of connectivity constraints. The approach is generic and can be used whenever all the relevant information about a "partial solution" can be encoded as an independent set of a specific linear matroid. We exemplify the approach on the STEINER TREE problem.

STEINER TREE
Input: An undirected graph G with a set of terminals $T \subseteq V(G)$ and a weight function $w : E(G) \to \mathbb{N}$.
Task: Find a subtree in G of minimum weight spanning all vertices of T.

5.4.1. 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

—$\cup_{t \in V(\mathbb{T})} X_t = V(G)$,
—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 $\mathbb{T}$ and is different than the root, and $X_t = \emptyset$.

Notice that, according to this 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 [Kloks 1994]. We use G_t to denote the graph induced by the vertex set $\cup_{t'} X_{t'}$, 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 $\mathbb{T}$.

5.4.2. STEINER TREE *Parameterized by Treewidth.* 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 the family of edge subsets such that every edge subset corresponds to an optimal solution. That is,

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

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 $E(H_t)$, $\{E' \subseteq E(H_t)\}$, that satisfies the following properties:

—either E' is a solution (i.e., the subgraph formed by this edge set is connected and contains all the terminal vertices) or
—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 a 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 $F(E^*)$ 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 $F(E^*)$ is to keep track for the vertices in C_i whether they are in the same connected component of G^*.
(3) We define $w(F(E^*)) = w(E^*)$.

Our description of the algorithm slightly deviates from the usual table lookup-based expositions of dynamic programming algorithms on graphs of bounded treewidth. 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, rather than keeping a table, 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. Here, $L_R = L \setminus L_t$. 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 3.7 for graphic matroids. More details are given in the proof of the following theorem, which is the main result of this section.

THEOREM 5.13. *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 $\mathcal{O}((1+2^{\omega+1})^{\mathbf{tw}}\mathbf{tw}^{\mathcal{O}(1)}n)$.*

PROOF. We first outline an algorithm with running time $\mathcal{O}((1+2^{\omega+1})^{\mathbf{tw}}\mathbf{tw}^{\mathcal{O}(1)}n^2)$ for a simple exposition. Later we point out how we can remove the extra factor of n at the cost of a factor polynomial in $\mathbf{tw}$.

For every node t of $\mathbb{T}$ and subset $Z \subseteq X_t$, we store a family of edge subsets $\widehat{\mathcal{S}}_t[Z]$ 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 $w(\hat{L}_t) \leq w(L_t)$, $\hat{L} = \hat{L}_t \cup L_R$ is a solution, and $\partial^t(\hat{L}) = Z$. Observe that since $w(\hat{L}_t) \leq w(L_t)$ 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 $\widehat{\mathcal{S}}_t[Z]$ that satisfies the correctness invariant, we use Theorem 3.7 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 $2^{|Z|}$. 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 $|\widehat{\mathcal{S}}_t[Z]| \leq 2^{|Z|}$.

The new ingredient of the dynamic programming algorithm for Steiner Tree is the use of Theorem 3.7 to compute $\widehat{\mathcal{S}}_t[Z]$ maintaining the size invariant. The next lemma shows how to implement it.

Lemma 5.14 (Shrinking Lemma). *Let t be a node of $\mathbb{T}$, and let $Z \subseteq X_t$ be a set of size k. Furthermore, let $\widehat{\mathcal{S}}_t[Z]$ be a family of edge subsets of H_t satisfying the correctness invariant. If $|\widehat{\mathcal{S}}_t[Z]| = \ell$, then in time $\mathcal{O}(2^{k(\omega-1)}k^{\mathcal{O}(1)}\ell \cdot n)$ 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 $K^t[Z]$.

Let $\widehat{\mathcal{S}}_t[Z] = \{E^t_1, \ldots, E^t_\ell\}$ and let $\mathcal{N} = \{F(E^t_1), \ldots, F(E^t_\ell)\}$ be the set of forests in $K^t[Z]$ corresponding to the edge subsets in $\widehat{\mathcal{S}}_t[Z]$. For $i \in \{1, \ldots, k-1\}$, let $\mathcal{N}_i$ be the family of forests of $\mathcal{N}$ with i edges. For each family $\mathcal{N}_i$, we apply Theorem 3.7 and compute its min $(k-1-i)$-representative. That is,

$$\widehat{\mathcal{N}}_i \subseteq^{k-1-i}_{minrep} \mathcal{N}_i.$$

Let $\widehat{\mathcal{S}}'_t[Z] \subseteq \widehat{\mathcal{S}}_t[Z]$ be such that for every $E^t_j \in \widehat{\mathcal{S}}'_t[Z]$ we have that $F(E^t_j) \in \cup_{i=1}^{k-1}\widehat{\mathcal{N}}_i$. By Theorem 3.7, $|\widehat{\mathcal{S}}'_t[Z]| \leq \sum_{i=1}^{k-1}\binom{k}{i} \leq 2^k$. Now we show that $\widehat{\mathcal{S}}'_t[Z]$ maintains the correctness invariant.

Let $L \in \mathscr{S}$ and let $L_t = E(H_t) \cap L$, $L_R = L \setminus L_t$ and $Z = \partial^t(L)$. Then there exists $E^t_j \in \widehat{\mathcal{S}}_t[Z]$ such that $w(E^t_j) \leq w(L_t)$, $\hat{L} = E^t_j \cup L_R$ is an optimal solution and $\partial^t(\hat{L}) = Z$. Consider the forest $F(E^t_j)$. Suppose its size is i, and then $F(E^t_j) \in \mathcal{N}_i$. Now let $F(L_R)$ be the forest corresponding to L_R with respect to the bag X_t. Since $\hat{L}$ is a solution, we have that $F(E^t_j) \cup F(L_R)$ is a spanning tree in $K^t[Z]$. Since $\widehat{\mathcal{N}}_i \subseteq^{k-1-i}_{minrep} \mathcal{N}_i$, we have that there exists a forest $F(E^t_h) \in \widehat{\mathcal{N}}_i$ such that $w(F(E^t_h)) \leq w(F(E^t_i))$ and $F(E^t_h) \cup F(L_R)$ is a spanning tree in $K^t[Z]$. Thus, we know that $E^t_h \cup L_R$ is an optimum solution and $E^t_h \in \widehat{\mathcal{S}}'_t[Z]$. This proves that $\widehat{\mathcal{S}}'_t[Z]$ maintains the invariant.

The running time to compute $\widehat{\mathcal{S}}_t[Z]$ is dominated by

$$\mathcal{O}\left(\sum_{i=1}^{k-1}\binom{k-1}{i}^{\omega-1}k^{\mathcal{O}(1)}\ell\right)=\mathcal{O}\big(2^{k(\omega-1)}k^{\mathcal{O}(1)}\ell\big).$$

For a given edge set we also need to compute the forest and that can take $\mathcal{O}(n)$ time. □

In our algorithm, the size of $\widehat{\mathcal{S}}_t[Z]$ can grow larger than $2^{|Z|}$ in intermediate steps but it will be at most $4^{|Z|}$ and thus we can use the Shrinking Lemma (Lemma 5.14) to reduce its size efficiently.

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 the 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 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 the 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 $Z\subseteq X_t$. Let $\mathcal{P}_v[Z]=\{Y\mid\emptyset\neq Y\subseteq\mathcal{E}_v[Z]\}$. 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. The idea is that for every edge subset in our family of partial solutions, we make several new partial solutions, one each for every subset of newly added edges. More formally, the new set of partial solutions is defined as follows:

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

Recall that for two families $\mathcal{A}$ and $\mathcal{B}$, we defined $\mathcal{A}\circ\mathcal{B}=\{A\cup B:A\in\mathcal{A}\wedge B\in\mathcal{B}\}$. Now we claim that $\widehat{\mathcal{S}}_t[Z]\subseteq\mathcal{S}_t$. Toward the proof we first show that $\widehat{\mathcal{S}}_{t'}[Z\cup\{v\}]\circ\mathcal{P}_v[Z]\subseteq\mathcal{S}_t$. Let $E'\in\widehat{\mathcal{S}}_{t'}[Z\cup\{v\}]\circ\mathcal{P}_v[Z]$. 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' that 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 $\hat{L}_{t'} \in \widehat{\mathcal{S}}_{t'}[Z']$ such that

$$\begin{aligned} L \in \mathscr{S} &\iff L_{t'} \uplus L_v \uplus L_R \in \mathscr{S} \\ &\iff \hat{L}_{t'} \uplus L_v \uplus L_R \in \mathscr{S} \end{aligned} \tag{7}$$

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 $\hat{L}_t \in \widehat{\mathcal{S}}_t[Z]$. Toward this just note that since $Z' = Z$ or $Z' = Z \cup \{v\}$, we have that $\widehat{\mathcal{S}}_t[Z]$ contains $\widehat{\mathcal{S}}_{t'}[Z'] \circ \{L_v\}$. By Equation (7), $\hat{L} \in \mathscr{S}$. Finally, we need to show that $\partial^t(\hat{L}) = Z$. Toward 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 are 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 to take the union of edge subsets of the families stored at nodes t_1 and t_2. This is exactly what we do. Let

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

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 $L_t = L_{t_1} \uplus L_{t_2}$.
(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

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

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 $\hat{L}_{t_1} \cup \hat{L}_{t_2} \in \widehat{\mathcal{S}}[Z]$. The previous 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 the correctness invariant.

Root Node r. Here, $X_r = \emptyset$. We go through all the solutions 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 previously in the desired running time by making use of Lemma 5.14. For our discussion, let us fix a node t and $Z \subseteq X_t$ of size k. While doing the dynamic programming algorithm from the base nodes to the root node, we always maintain the size invariant. That is, $|\widehat{\mathcal{S}}_t[Z]| \leq 2^k$.

Base Node t. Trivially, in this case we have $|\widehat{\mathcal{S}}_t[Z]| \leq 2^k$.

Introduce Node t with Child t'. Here, we have that $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t'}[Z \setminus \{v\}]$ and thus $|\widehat{\mathcal{S}}_t[Z]| = |\widehat{\mathcal{S}}_{t'}[Z \setminus \{v\}]| \leq 2^{k-1} \leq 2^k$.

Forget Node t with Child t'. In this case,

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

Observe that

$$\begin{aligned} \left|\widehat{\mathcal{S}}_t[Z]\right| &\leq \left|\widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \circ \mathcal{P}_v[Z]\right| + \left|\left\{A \in \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] : A \in \mathcal{S}_t\right\}\right| + \left|\widehat{\mathcal{S}}_{t'}[Z]\right| \\ &\leq \left(\sum_{i=1}^{k} \binom{k}{i} 2^{k+1}\right) + 2^{k+1} + 2^k = \mathcal{O}(4^k). \end{aligned}$$

It can happen in this case that the size of $\widehat{\mathcal{S}}_t[Z]$ is larger than 2^k and thus we need to reduce the size of the family. We apply Lemma 5.14 and obtain $\widehat{\mathcal{S}}'_t[Z]$, which maintains the correctness and size invariants. We update $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}'_t[Z]$.

The running time to compute $\widehat{\mathcal{S}}_t$ (i.e., across all subsets of X_t) is

$$\mathcal{O}\left(\sum_{i=1}^{\mathbf{tw}+1} \binom{\mathbf{tw}+1}{i} 2^{i(\omega-1)} 4^i \cdot \mathbf{tw}^{\mathcal{O}(1)} n\right) = \mathcal{O}\left((1+2^{\omega+1})^{\mathbf{tw}} \cdot \mathbf{tw}^{\mathcal{O}(1)} n\right).$$

Join Node t with Two Children t_1 and t_2. Here we defined

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

The size of $\widehat{\mathcal{S}}_t[Z]$ is $2^k \cdot 2^k = 4^k$. Now, we apply Lemma 5.14 and obtain $\widehat{\mathcal{S}}'_t[Z]$, which 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

$$\mathcal{O}\left(\sum_{i=1}^{\mathbf{tw}+1} \binom{\mathbf{tw}+1}{i} 4^i 2^{i(\omega-1)} \cdot \mathbf{tw}^{\mathcal{O}(1)} n\right) = \mathcal{O}\left((1+2^{\omega+1})^{\mathbf{tw}} \cdot \mathbf{tw}^{\mathcal{O}(1)} n\right).$$

Thus, the whole algorithm takes time $\mathcal{O}((1+2^{\omega+1})^{\mathbf{tw}} \cdot \mathbf{tw}^{\mathcal{O}(1)} \cdot n^2)$ as the number of nodes in a nice tree decomposition is upper bounded by $\mathcal{O}(n)$. 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 $\mathbf{tw}^{\mathcal{O}(1)}$. This will lead to an algorithm with the claimed running time. The last remark we would like to make is that one can do better at **forget node** by forgetting a single edge at a time. However, we did not try to optimize this, as the running time to compute the family of partial solutions at **join node** is the most expensive operation. This completes the proof. □

The approach of Theorem 5.13 can be used to obtain single-exponential algorithms parameterized by the treewidth of an input graph for several other connectivity problems such as Hamiltonian Cycle, Feedback Vertex Set, and Connected Dominated Set. For all these problems, checking whether two partial solutions can be glued together to form a global solution can be checked by testing independence in a specific graphic matroid. We believe that there exist interesting problems where this check corresponds to testing independence in a different class of linear matroids.

5.5. Path, Trees, and Subgraph Isomorphism

In this section, we outline algorithms for k-Path, k-Tree, and k-Subgraph Isomorphism using representative families. All results in this section are based on computing representative families with respect to uniform matroids.

*5.5.1. k-*PATH. The problem we study in this section is as follows:

k-PATH	**Parameter:** k
Input: An undirected n-vertex and m-edge graph G and a positive integer k.	
Question: Does there exist a simple path of length k in G?	

We start by modifying the graph slightly. We add a new vertex, say, s, not present in $V(G)$ to G by making it adjacent to every vertex in $V(G)$. Let the modified graph be called G'. It is clear that G has a path of length k if and only if G' has a path of length $k+1$ starting from s. For ease of presentation, we rename G' to G and the objective is to find a path of length $k+1$ starting from s. Let $M = (E, \mathcal{I})$ be a uniform matroid $U_{n,k+2}$, where $E = V(G)$ and $\mathcal{I} = \{S \subseteq V(G) \,|\, |S| \leq k+2\}$. In this section, whenever we speak about independent sets, we mean independence with respect to the uniform matroid $U_{n,k+2}$ defined earlier. For a given pair of vertices $s, v \in V(G)$, recall that we defined

$$\mathcal{P}^i_{sv} = \big\{X \big| X \subseteq V(G),\ v, s \in X,\ |X| = i \text{ and there is a path from } s \text{ to } v \text{ of length } i \text{ in } G \text{ with all the vertices belonging to } X.\big\}$$

The problem can be reformulated to asking whether there exists $v \in V(G)$ such that $\mathcal{P}^{k+2}_{sv}$ is nonempty. Our algorithm will check whether $\mathcal{P}^{k+2}_{uv}$ is nonempty by computing $\widehat{\mathcal{P}}^{k+2}_{sv} \subseteq^0_{rep} \mathcal{P}^{k+2}_{sv}$ and checking whether $\widehat{\mathcal{P}}^{k+2}_{sv}$ is nonempty. The correctness of this algorithm is as follows. If $\mathcal{P}^{k+2}_{sv}$ is nonempty, then $\mathcal{P}^{k+2}_{sv}$ contains some set A that does not intersect the empty set $\emptyset$. But then $\widehat{\mathcal{P}}^{k+2}_{sv} \subseteq^0_{rep} \mathcal{P}^{k+2}_{sv}$ must also contain a set that does not intersect with $\emptyset$, and hence $\widehat{\mathcal{P}}^{k+2}_{sv}$ must be nonempty as well. Thus, having computed the representative families $\widehat{\mathcal{P}}^{k+2}_{sv}$, all we need to do is to check whether there is a vertex v such that $\widehat{\mathcal{P}}^{k+2}_{sv}$ is nonempty. All that remains is an algorithm that computes the representative families $\widehat{\mathcal{P}}^{k+2}_{sv} \subseteq^0_{rep} \mathcal{P}^{k+2}_{sv}$ for all $v \in V(G) \setminus \{s\}$.

Now, using Lemma 5.4 (by setting $\ell = p = k+2$ and replacing each edge by two arcs in opposite directions), we compute $\widehat{\mathcal{P}}^{k+2}_{sv} \subseteq^0_{rep} \mathcal{P}^{k+2}_{sv}$ for all $v \in V(G) \setminus \{s\}$ in time

$$2^{o(k)} \cdot m \log n \cdot \max_{i \in [k+2]} \left\{ \left(\frac{2(k+2)-i}{i} \right)^i \left(\frac{2(k+2)-i}{2(k+2)-2i} \right)^{2(k+2)-2i} \right\}.$$

Simple calculus shows that the running time is maximized for $i = (1 - \frac{1}{\sqrt{5}})(k+2)$, and thus the running time to compute $\widehat{\mathcal{P}}^{k+2}_{sv} \subseteq^0_{rep} \mathcal{P}^{k+2}_{sv}$ for all $v \in V(G) \setminus \{s\}$ together is upper bounded by $\phi^{2k+o(k)} m \log^2 n = \mathcal{O}(2.619^{k+o(k)} m \log n)$, where ϕ is the golden ratio $\frac{1+\sqrt{5}}{2}$. Furthermore, in the same time every set in $\widehat{\mathcal{P}}^p_{sv}$ can be ordered in a way that it corresponds to an undirected path in G. A graph G has a path of length $k+1$ starting from s if and only if for some $v \in V(G) \setminus \{s\}$, we have that $\widehat{\mathcal{P}}^{k+2}_{sv} \neq \emptyset$. Thus, the running time of this algorithm is upper bounded by $\mathcal{O}(2.619^{k+o(k)} m \log n)$. Let us remark that almost the same arguments show that the version of the problem on directed graphs is solvable within the same running time. However, on undirected graphs, we can speed up the algorithm slightly by using the following standard trick. We need the following result.

PROPOSITION 5.15 (BODLAENDER [1993]). *There exists an algorithm that, given a graph G and an integer k, in time $\mathcal{O}(k^2 n)$ either finds a simple path of length $\geq k$ or computes a DFS (depth-first search) tree rooted at some vertex of G of depth at most k.*

We first apply Proposition 5.15 and in time $\mathcal{O}(k^2n)$ either find a simple path of length $\geq k$ in G or compute a DFS tree of G of depth at most k. In the former case, we simply output the same path. In the latter case, since all the root-to-leaf paths are upper bounded by k and there are no cross-edges in a DFS tree, we have that the number of edges in G is upper bounded by $\mathcal{O}(k^2n)$. Now on this G we apply the representative-family-based algorithm described previously. This results in the following theorem.

THEOREM 5.16. *k-PATH can be solved in time $\mathcal{O}(2.619^{k+o(k)}n \log n)$.*

Our algorithm for k-PATH can be used to solve the weighted version of the problem, that is, SHORT CHEAP TOUR. In this problem, a graph G with maximum edge cost W is given, and the objective is to find a path of length at least k where the total sum of costs on the edges is minimized.

THEOREM 5.17. SHORT CHEAP TOUR *can be solved in time $\mathcal{O}(2.619^{k+o(k)}n^{\mathcal{O}(1)} \log W)$.*

5.5.2. k-TREE and k-SUBGRAPH ISOMORPHISM. In this section, we consider the following problem:

k-TREE	**Parameter:** k
Input: An undirected n-vertex, an m-edge graph G, and a tree T on k vertices.	
Question: Does G contains a subgraph isomorphic to T?	

We design an algorithm for k-TREE using the method of representative families. The algorithm for k-TREE is more involved than for k-PATH. The reason for that is due to the fact that paths pose perfectly balanced separators of size one, while trees do not. We select a leaf r of T and root the tree at r. For vertices $x, y \in V(T)$, we say that $y \leq x$ if x lies on the path from y to r in T (if $x = r$, we also say that $y \leq x$). For a set C of vertices in T, we will say that $x \preceq_C y$ if $x \leq y$ and there is no $z \in C$ such that $x \leq z$ and $z \leq y$. For a pair x, y of vertices such that $y \leq x$ in T, we define

$$C^{xy} = \begin{cases} \emptyset & \text{if } xy \in E(T), \\ \text{The unique component } C \text{ of } T \setminus \{x, y\} \text{ such that } N(C) = \{x, y\} & \text{otherwise.} \end{cases}$$

We also define $T^{uv} = T[C^{uv} \cup \{u, v\}]$. We start by making a few simple observations about sets of vertices in trees.

LEMMA 5.18. *For any tree T, a pair $\{x, y\}$ of vertices in $V(T)$, and an integer $c \geq 1$, there exists a set W of vertices such that $\{x, y\} \subseteq W$, $|W| = \mathcal{O}(c)$, and every connected component U of $T \setminus W$ satisfies $|U| \leq \frac{|V(T)|}{c}$ and $|N(U)| \leq 2$.*

PROOF. We first find a set W_1 of size at most c such that every connected component U of $T \setminus W_1$ satisfies $|U| \leq \frac{|V(T)|}{c}$. Start with $W_1 = \emptyset$ and select a lowermost vertex $u \in V(T)$ such that the subtree rooted at u has at least $\frac{|V(T)|}{c}$ vertices. Add u to W_1 and remove the subtree rooted at u from T. The process must stop after c iterations since each iteration removes $\frac{|V(T)|}{c}$ vertices of T. Each component U of $T \setminus W_1$ satisfies $|U| \leq \frac{|V(T)|}{c}$ because (1) whenever a vertex u is added to W_1, all components below u have size strictly less than $\frac{|V(T)|}{c}$, and (2) when the process ends, the subtree rooted at r has size at most $|U| \leq \frac{|V(T)|}{c}$. Now, insert x and y into W_1 as well.

We build W from W_1 by taking the *least common ancestor closure* of W_1; start with $W = W_1$ and as long as there exist two vertices u and v in W such that their least common ancestor w is not in W, add w to W. Standard counting arguments on trees

imply that this process will never increase the size of W by more than a factor of 2; hence, $|W| \leq 2|W_1| = O(c)$.

We claim that every connected component U of $T \setminus W$ satisfies $|N(U)| \leq 2$. Suppose not and let u be the vertex of U closest to the root. Since $|N(U)| > 2$, at least two vertices v and w in $N(U)$ are descendants of u. Since U is connected, v and w can't be descendants of each other, but then the least common ancestor of v and w is in U, contradicting the construction of W. □

Let us remark that in the following observation, if the root of a tree is of degree 1, then we also count it as a leaf.

OBSERVATION 5.1. *For any tree T, set $W \subseteq V(T)$ and component U of $T \setminus W$ such that $|N(U)| = 1$, U contains a leaf of T.*

PROOF. $T[U \cup N(U)]$ is a tree on at least two vertices and hence it has at least two leaves. At most one of these leaves is in $N(U)$; the other one is also a leaf of T. □

LEMMA 5.19. *Let $W \subseteq V(T)$ be a set of vertices such that for every pair of vertices in W, their least common ancestor is also in W. Let X be a set containing one leaf of T from each connected component U of $T \setminus W$ such that $|N(U)| = 1$. Then, for every connected component U such that $|N(U)| = 1$, there exist $x \in W$, $y \in X$ such that $U = C^{xy} \cup \{y\}$. For every other connected component U, there exist $x, y \in W$ such that $U = C^{xy}$.*

PROOF. It follows from the argument at the end of the proof of Lemma 5.18 that every component U of $T \setminus W$ satisfies $|N(U)| \leq 2$. If $|N(U)| = 2$, let $N(U) = \{x, y\}$. We have that $x \leq y$ or $y \leq x$ since the least common ancestor of x and y cannot be in U and would therefore be in $N(U)$, contradicting $|N(U)| = 2$. Without loss of generality, $y \leq x$. But then $U = C^{xy}$. If $N(U) = 1$, let $N(U) = \{x\}$. By Observation 5.1, U contains a leaf y of T. Then $U = C^{xy} \cup \{y\}$. □

Given two graphs F and H, a graph *homomorphism* from F to H is a map f from $V(F)$ to $V(H)$, that is, $f : V(F) \to V(H)$, such that if $uv \in E(F)$, then $f(u)f(v) \in E(H)$. Furthermore, when the map f is injective, f is called a *subgraph isomorphism*. For every $x, y \in V(T)$ such that $y \leq x$, and every u, v in $V(G)$, we define

$$\mathcal{F}_{uv}^{xy} = \left\{ F \in \binom{V(G) \setminus \{u, v\}}{|C^{xy}|} : \exists \text{ subgraph isomorphism } f \right.$$
$$\left. \text{from } T^{xy} \text{ to } G[F \cup \{u, v\}] \text{ such that } f(x) = u \text{ and } f(y) = v \right\}.$$

Let us remember that for a set X and a family $\mathcal{A}$, we use $\mathcal{A}+X$ to denote $\{A \cup X : A \in \mathcal{A}\}$. For every $x, y \in V(T)$ such that $y \leq x$, and every u in $V(G)$, we define

$$\mathcal{F}_{u*}^{xy} = \bigcup_{v \in V(G) \setminus \{u\}} \mathcal{F}_{uv}^{xy} + \{v\}. \tag{8}$$

We recall that r is a leaf of T. In order to solve the problem, it is sufficient to select an arbitrary leaf ℓ of T and determine whether there exists a $u \in V(G)$ such that the family $\mathcal{F}_{u*}^{r\ell}$ is nonempty. We show that the collections of families $\{\mathcal{F}_{uv}^{xy}\}$ and $\{\mathcal{F}_{u*}^{xy}\}$ satisfy a recurrence relation. We will then exploit this recurrence relation to get a fast algorithm for k-TREE.

LEMMA 5.20. *For every $x, y \in V(T)$ such that $y \leq x$, every $\widehat{W} = W \cup \{x, y\}$ where $W \subseteq C^{xy}$ such that for every pair of vertices in $\widehat{W}$ their least common ancestor is also in $\widehat{W}$, and every $X \subseteq C^{xy} \setminus W$ such that X contains exactly one leaf of T in each connected*

component U of $T^{xy} \setminus \widehat{W}$ with $|N(U)| = 1$, the following recurrence holds:

$$\mathcal{F}^{xy}_{uv} = \bigcup_{\substack{g:\widehat{W}\to V(G)\\ g(x)=u \wedge g(y)=v}} \left[\left(\dot{\prod_{\substack{x',y'\in\widehat{W}\\ y'\preceq_{\widehat{W}} x'}}} \mathcal{F}^{x'y'}_{g(x')g(y')} \bullet \dot{\prod_{\substack{x'\in\widehat{W},\, y'\in X\\ y'\preceq_{\widehat{W}} x'}}} \mathcal{F}^{x'y'}_{g(x')*} \right) + g(W) \right]. \tag{9}$$

Here the union goes over all $O(n^{|W|})$ injective maps g from $\widehat{W}$ to $V(G)$ such that $g(x) = u$ and $g(y) = v$, and by $g(W)$ we mean $\{g(c) : c \in W\}$.

Proof. For the $\subseteq$ direction of the equality, consider any subgraph isomorphism f from T^{xy} to $V(G)$ such that $f(x) = u$ and $f(y) = v$. Let g be the restriction of f to W. The map f can be considered as a collection of subgraph isomorphisms with one isomorphism for each $x', y' \in \widehat{W}$ such that $y' \preceq_{\widehat{W}} x$ from $T^{x'y'}$ to G such that $f(x') = g(x')$ and $f(y') = g(y')$, and one isomorphism for each $x' \in \widehat{W}, y' \in X$ such that $y' \preceq_{\widehat{W}} x$ from $T^{x'y'}$ to G such that $f(x') = g(x')$. Taking the union of the ranges of each of the small subgraph isomorphisms clearly gives the range of f. Here we used Lemma 5.19 to argue that for every connected component U of $T^{xy} \setminus \widehat{W}$, we have that $T[U \cup N(U)]$ is in fact of the form $T^{x'y'}$ for some x', y'.

For the reverse direction, take any collection of subgraph isomorphisms with one isomorphism f for each $x', y' \in \widehat{W}$ such that $y' \preceq_{\widehat{W}} x$ from $T^{x'y'}$ to G such that $f(x') = g(x')$ and $f(y') = g(y')$, and one isomorphism for each $x' \in \widehat{W}, y' \in X$ such that $y' \preceq_{\widehat{W}} x$ from $T^{x'y'}$ to G such that $f(x') = g(x')$, such that the range of all of these subgraph isomorphisms are pairwise disjoint (except on vertices in $\widehat{W}$). Since all of these subgraph isomorphisms agree on the set W, they can be glued together to a subgraph isomorphism from T^{xy} to G. □

Our goal is to compute for every $x, y \in V(T)$ such that $y \le x$ and $u, v \in V(G)$ a family $\hat{\mathcal{F}}^{xy}_{uv}$ such that $\hat{\mathcal{F}}^{xy}_{uv} \subseteq^{k-|C^{xy}|}_{rep} \mathcal{F}^{xy}_{uv}$ and for every $x, y \in V(T)$ such that $y \le x$ and $u \in V(G)$ a family $\hat{\mathcal{F}}^{xy}_{u*}$ such that $\hat{\mathcal{F}}^{xy}_{u*} \subseteq^{k-|C^{xy}|-1}_{rep} \mathcal{F}^{xy}_{u*}$. We will also maintain the following size invariants:

$$|\hat{\mathcal{F}}^{xy}_{uv}| \le \left(\frac{2k-|C^{xy}|}{|C^{xy}|}\right)^{|C^{xy}|} \left(\frac{2k-|C^{xy}|}{2k-2|C^{xy}|}\right)^{k-|C^{xy}|} 2^{o(k)} \tag{10}$$

$$|\hat{\mathcal{F}}^{xy}_{u*}| \le \left(\frac{2k-|C^{xy}|-1}{|C^{xy}|+1}\right)^{|C^{xy}|+1} \left(\frac{2k-|C^{xy}|-1}{2k-2|C^{xy}|-2}\right)^{k-|C^{xy}|-1} 2^{o(k)}. \tag{11}$$

Let the right-hand side of Equation (10) be s_{xy} and the right-hand side of Equation (11) be s^*_{xy}. We first compute such families $\hat{\mathcal{F}}^{xy}_{uv}$ for all $x, y \in V(T)$ such that $y \le x$ and $xy \in E(T)$. Observe that in this case we have

$$\mathcal{F}^{xy}_{uv} = \begin{cases} \{\emptyset\} & \text{if } uv \in E(G), \\ \emptyset & \text{if } uv \notin E(G). \end{cases}$$

For each $x, y \in V(T)$ such that $y \le x$ and $xy \in E(T)$ and every $u, v \in V(G)$, we set $\hat{\mathcal{F}}^{xy}_{uv} = \mathcal{F}^{xy}_{uv}$. We can now compute $\hat{\mathcal{F}}^{xy}_{u*}$ for every $x, y \in V(T)$ such that $y \le x$ and $xy \in E(T)$ and every $u \in V(G)$ by applying Equation (8). Clearly the computed families are within the required size bounds.

We now show how to compute a family $\hat{\mathcal{F}}^{xy}_{uv}$ of size s_{xy} for every $x, y \in V(T)$ such that $y \le x$ and $u, v \in V(G)$ and $|C^{xy}| = t$, assuming that the families $\hat{\mathcal{F}}^{xy}_{uv}$ and $\hat{\mathcal{F}}^{xy}_{u*}$ have been

computed for every $x, y \in V(T)$ such that $y \leq x$ and $u, v \in V(G)$ and $|C^{xy}| < t$. We also assume that for each family $\hat{\mathcal{F}}^{xy}_{uv}$ that has been computed, $|\hat{\mathcal{F}}^{xy}_{uv}| \leq s_{xy}$. Similarly, we assume that for each family $\hat{\mathcal{F}}^{xy}_{u*}$ that has been computed, $|\hat{\mathcal{F}}^{xy}_{u*}| \leq s^*_{xy}$.

We fix a constant c whose value will be decided later. First, apply Lemma 5.18 on T^{xy}, vertex pair $\{x, y\}$, and constant c and obtain a set $\widehat{W}$ such that $\{x, y\} \subseteq \widehat{W}$ and every connected component U of $T \setminus \widehat{W}$ satisfies $|U| \leq \frac{|V(T)|}{c}$ and $|N(U)| \leq 2$. Select a set $X \subseteq V(T^{x,y}) \setminus \widehat{W}$ such that each connected component U of $T \setminus \widehat{W}$ with $|N(U)| = 1$ contains exactly one leaf, which is in X. Now, set $W = \widehat{W} \setminus \{x, y\}$ and consider Equation (9) for $\hat{\mathcal{F}}^{xy}_{uv}$ for this choice of x, y, W and X. Define

$$\tilde{\mathcal{F}}^{xy}_{uv} = \bigcup_{\substack{g:\widehat{W}\to V(G)\\ g(x)=u \wedge g(y)=v}} \left[\left(\dot{\prod_{\substack{x',y' \in \widehat{W}\\ y' \preceq_{\widehat{W}} x'}}} \hat{\mathcal{F}}^{x'y'}_{g(x')g(y')} \bullet \dot{\prod_{\substack{x' \in \widehat{W},\, y' \in X\\ y' \preceq_{\widehat{W}} x'}}} \hat{\mathcal{F}}^{x'y'}_{g(x')*} \right) + g(W) \right]. \tag{12}$$

Lemma 5.20 together with Lemmata 3.4 and 3.5 directly imply that $\tilde{\mathcal{F}}^{xy}_{uv} \subseteq^{k-|C^{xy}|}_{rep} \mathcal{F}^{xy}_{uv}$. Furthermore, each family on the right-hand side of Equation (12) has already been computed, since $C^{x'y'} \subset C^{xy}$ and so $|C^{x'y'}| < t$. For a fixed injective map $g : W \to V(G)$, we define

$$\tilde{\mathcal{F}}^{xy}_{g} = \left(\dot{\prod_{\substack{x',y' \in \widehat{W}\\ y' \preceq_{\widehat{W}} x'}}} \hat{\mathcal{F}}^{x'y'}_{g(x')g(y')} \bullet \dot{\prod_{\substack{x' \in \widehat{W},\, y' \in X\\ y' \preceq_{\widehat{W}} x'}}} \hat{\mathcal{F}}^{x'y'}_{g(x')*} \right) + g(W). \tag{13}$$

It follows directly from the definition of $\tilde{\mathcal{F}}^{xy}_{uv}$ and $\tilde{\mathcal{F}}^{xy}_{g}$ that

$$\tilde{\mathcal{F}}^{xy}_{uv} = \bigcup_{\substack{g:\widehat{W}\to V(G)\\ g(x)=u \wedge g(y)=v}} \tilde{\mathcal{F}}^{xy}_{g}.$$

Our goal is to compute a family $\hat{\mathcal{F}}^{xy}_{uv} \subseteq^{k-|C^{xy}|}_{rep} \tilde{\mathcal{F}}^{xy}_{uv}$ such that $|\hat{\mathcal{F}}^{xy}_{uv}| \leq s_{xy}$. Lemma 3.3 then implies that $\hat{\mathcal{F}}^{xy}_{uv} \subseteq^{k-|C^{xy}|}_{rep} \mathcal{F}^{xy}_{uv}$. To that end, we define the function reduce. Given a family $\mathcal{F}$ of sets of size p, the function reduce will run the algorithm of Theorem 4.15 on $\mathcal{F}$ with $x = \frac{p}{2k-p}$ and produce a family of size $(\frac{2k-p}{p})^p (\frac{2k-p}{2k-2p})^{k-p} 2^{o(k)}$ that $k-p$ represents $\mathcal{F}$.

We will compute for each $g : \widehat{W} \to V(G)$ such that $g(x) = u$ and $g(y) = v$ a family $\hat{\mathcal{F}}^{xy}_{g}$ of size at most s_{xy} such that $\hat{\mathcal{F}}^{xy}_{g} \subseteq^{k-|C^{xy}|}_{rep} \tilde{\mathcal{F}}^{xy}_{g}$. We will then set

$$\hat{\mathcal{F}}^{xy}_{uv} = \mathsf{reduce}\left(\bigcup_{\substack{g:\widehat{W}\to V(G)\\ g(x)=u \wedge g(y)=v}} \hat{\mathcal{F}}^{xy}_{g} \right). \tag{14}$$

To compute $\hat{\mathcal{F}}^{xy}_{g}$, inspect Equation (13). Equation (13) shows that $\tilde{\mathcal{F}}^{xy}_{g}$ basically is a long chain of $\bullet$ operations, specifically

$$\tilde{\mathcal{F}}^{xy}_{g} = \left(\hat{F}_1 \bullet \hat{F}_2 \bullet \hat{F}_3 \cdots \bullet \hat{F}_\ell\right) + g(W). \tag{15}$$

We define (and compute) $\hat{\mathcal{F}}^{xy}_{g}$ as follows:

$$\hat{\mathcal{F}}^{xy}_{g} = \mathsf{reduce}\big(\mathsf{reduce}\big(\ldots \mathsf{reduce}\big(\mathsf{reduce}\big(\hat{F}_1 \bullet \hat{F}_2\big) \bullet \hat{F}_3\big) \bullet \ldots\big) \bullet \hat{F}_\ell\big) + g(W). \tag{16}$$

$\hat{\mathcal{F}}_g^{xy} \subseteq_{rep}^{k-|C^{xy}|} \tilde{\mathcal{F}}_g^{xy}$ and thus also $\hat{\mathcal{F}}_{uv}^{xy} \subseteq_{rep}^{k-|C^{xy}|} \tilde{\mathcal{F}}_{uv}^{xy} \subseteq_{rep}^{k-|C^{xy}|} \mathcal{F}_{uv}^{xy}$ follow from Lemma 3.5 and Theorem 4.15. Since the last operation we do in the construction of $\hat{\mathcal{F}}_{uv}^{xy}$ is a call to reduce, $|\hat{\mathcal{F}}_{uv}^{xy}| \leq s_{xy}$ follows from Theorem 4.15. To conclude the computation, we set

$$\tilde{\mathcal{F}}_{u*}^{xy} = \mathsf{reduce}\left(\bigcup_{v \in V(G)\setminus\{u\}} \hat{\mathcal{F}}_{uv}^{xy} + \{v\}\right). \tag{17}$$

Lemma 3.5 and Theorem 4.15 imply that $\tilde{\mathcal{F}}_{u*}^{xy} \subseteq_{rep}^{k-|C^{xy}|-1} \mathcal{F}_{u*}^{xy}$ and that $|\hat{\mathcal{F}}_{u*}^{xy}| \leq s_{xy}^*$.

The algorithm computes the families $\hat{\mathcal{F}}_{u*}^{xy}$ and $\hat{\mathcal{F}}_{uv}^{xy}$ for every $x, y \in V(T)$ such that $y \leq x$. It then selects an arbitrary leaf ℓ of T and checks whether there exists a $u \in V(G)$ such that the family $\hat{\mathcal{F}}_{u*}^{r\ell}$ is nonempty. Since $\hat{\mathcal{F}}_{u*}^{r\ell} \subseteq_{rep}^{0} \mathcal{F}_{u*}^{r\ell}$, there is a nonempty $\mathcal{F}_{u*}^{r\ell}$ if and only if there is a nonempty $\hat{\mathcal{F}}_{u*}^{r\ell}$. Thus, the algorithm can answer that there is a subgraph isomorphism from T to G if some $\hat{\mathcal{F}}_{u*}^{r\ell}$ is nonempty, and that no such subgraph isomorphism exists otherwise.

It remains to bound the running time of the algorithm. Up to polynomial factors, the running time of the algorithm is dominated by the computation of $\hat{\mathcal{F}}_{uv}^{xy}$. This computation consists of $n^{\mathcal{O}(|\widehat{W}|)}$ independent computations of the families $\hat{\mathcal{F}}_g^{xy}$. Each computation of the family $\hat{\mathcal{F}}_g^{xy}$ consists of at most k repeated applications of the operation

$$\hat{\mathcal{F}}^{i+1} = \mathsf{reduce}(\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1}).$$

Here $\mathcal{F}^i$ is a family of sets of size p, and so $|\mathcal{F}^i| \leq (\frac{2k-p}{p})^p (\frac{2k-p}{2k-2p})^{k-p} 2^{o(k)} \log n$. On the other hand, $\hat{\mathcal{F}}_{i+1}$ is a family of sets of size $p' \leq \frac{k}{c}$ since we used Lemma 5.18 to construct $\widehat{W}$. Thus,

$$\begin{aligned}
|\hat{\mathcal{F}}_{i+1}| &\leq \left(\frac{2k-p'}{p'}\right)^{p'} \left(\frac{2k-p'}{2k-2p'}\right)^{k-p'} 2^{o(k)} \\
&\leq \left(\frac{2k}{p'}\right)^{p'} \left(\frac{2k}{2k-2p'}\right)^{k-p'} 2^{o(k)} \\
&\leq \binom{k}{p'} \cdot 2^{p'} \cdot 2^{o(k)} \\
&\leq \binom{k}{k/c} \cdot 2^{k/c} \cdot 2^{o(k)} \\
&\leq 2^{(\varepsilon+1/c)k} \cdot 2^{o(k)}.
\end{aligned}$$

Thus, $|\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1}| \leq (\frac{2k-p}{p})^p (\frac{2k-p}{2k-2p})^{k-p} 2^{(\varepsilon+1/c)k+o(k)}$. Hence, when we apply Theorem 4.15 with $x = \frac{p+p'}{2k-p-p'}$ to compute $\mathsf{reduce}(\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1})$, this takes time

$$\begin{aligned}
&|\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1}| \left(\frac{2k-p-p'}{2k-2p-2p'}\right)^{k-p-p'} 2^{o(k)} \log n \\
&\quad \leq |\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1}| \left(\frac{2k-p}{2k-2p}\right)^{k-p} \left(\frac{2k-2p}{2k-2p-2p'}\right)^{k-p-p'} 2^{o(k)} \log n
\end{aligned}$$

$$\leq |\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1}| \left(\frac{2k-p}{2k-2p}\right)^{k-p} \left(1+\frac{p'}{k-p-p'}\right)^{k-p-p'} 2^{o(k)} \log n$$

$$\leq |\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1}| \left(\frac{2k-p}{2k-2p}\right)^{k-p} e^{p'} 2^{o(k)} \log n$$

$$\leq \left(\frac{2k-p}{p}\right)^{p} \left(\frac{2k-p}{2k-2p}\right)^{2k-2p} 2^{(\varepsilon+3/c)k+o(k)} \log n.$$

Since there are $n^{\mathcal{O}(|\widehat{W}|)}$ (which is equal to $n^{\mathcal{O}(c)}$, where c is a constant) independent computations of the families $\hat{\mathcal{F}}_g^{xy}$, the total running time is upper bounded by

$$\left(\frac{2k-p}{p}\right)^{p} \left(\frac{2k-p}{2k-2p}\right)^{2k-2p} 2^{(\varepsilon+3/c)k+o(k)} n^{\mathcal{O}(1)}.$$

The maximum value of $(\frac{2k-p}{p})^p (\frac{2k-p}{2k-2p})^{2k-2p}$ is when $p = (1-\frac{1}{\sqrt{5}})k$ and the maximum value is ϕ^{2k}, where ϕ is the golden ratio $\frac{1+\sqrt{5}}{2}$. Now we can choose the value of c in such a way that $\varepsilon + 3/c$ is small enough and the aforementioned running time is bounded by $2.619^{k+o(k)} n^{\mathcal{O}(1)}$. This yields the following theorem.

THEOREM 5.21. *k-TREE can be solved in time $2.619^{k+o(k)} n^{\mathcal{O}(1)}$.*

The algorithm for k-TREE can be generalized to k-SUBGRAPH ISOMORPHISM for the case when the pattern graph F has treewidth at most t. Toward this, we need a result analogous to Lemma 5.18 for trees, which can be proved using the separation properties of graphs of treewidth at most t. This will lead to an algorithm with running time $2.619^{k+o(k)} \cdot n^{\mathcal{O}(t)}$.

5.6. Other Applications

Marx [2009] gave algorithms for several problems based on matroid optimization. The main theorem in his work is Theorem 1.1 [Marx 2009] on which most applications of Marx [2009] are based. The proof of the theorem uses an algorithm to find representative families as a black box. Applying our algorithm (Theorem 1.1 of this article) instead gives an improved version of Theorem 1.1 of Marx [2009].

PROPOSITION 5.22. *Let $M = (E, \mathcal{I})$ be a linear matroid where the ground set is partitioned into blocks of size ℓ. Given a linear representation A_M of M, it can be determined in $\mathcal{O}(2^{\omega k\ell} ||A_M||^{\mathcal{O}(1)})$ randomized time whether there is an independent set that is the union of k blocks. ($||A_M||$ denotes the length of A_M in the input.)*

Finally, we mention another application from Marx [2009] that we believe could be useful to obtain single exponential time parameterized and exact algorithms:

ℓ-MATROID INTERSECTION	**Parameter:** k
Input: Let $M_1 = (E, \mathcal{I}_1), \ldots, M_1 = (E, \mathcal{I}_\ell)$ be matroids on the same ground set E given by their representations $A_{M_1}, \ldots, A_{M_\ell}$ over the same field $\mathbb{F}$ and a positive integer k.	
Question: Does there exist k element set that is independent in each M_i ($X \in \mathcal{I}_1 \cap \cdots \cap \mathcal{I}_\ell$)?	

Using Theorem 1.1 of [Marx 2009], Marx gave a randomized algorithm for ℓ-MATROID INTERSECTION. By using Proposition 5.22 instead, we get the following result.

PROPOSITION 5.23. *ℓ-MATROID INTERSECTION can be solved in $\mathcal{O}(2^{\omega k\ell}||A_M||^{\mathcal{O}(1)})$ randomized time.*

6. CONCLUSION AND RECENT DEVELOPMENTS

In this article, we gave an efficient algorithm for computing a representative family of a family of independent sets in a linear matroid. For the special case where the underlying matroid is uniform, we developed an even faster algorithm. We also showed interesting links between representative families of matroids and the design of single-exponential parameterized and exact exponential algorithms. We believe that these connections have a potential for a wide range of applications. This work opens up an interesting avenue for further research, and we list some of the natural open problems here:

—What is the best possible running time of an algorithm that computes a q-representative family of size at most $\binom{p+q}{p}$ for a p-family $\mathcal{F}$ of independent sets of a linear matroid? Does an algorithm with linear dependence of the running time on $|\mathcal{F}|$ exist, or is it possible to prove superlinear lower bounds?
—It would be interesting to find faster algorithms even for special classes of linear matroids. Uniform matroids and graphic matroids are especially interesting in this regard.
—Finally, the only matroids we used in our algorithmic applications were graphic, uniform, and partition matroids. It would be interesting to see what kinds of applications can be handled by other kinds of matroids.

The results and methods from the preliminary conference version of this article have already been utilized to obtain several deterministic parameterized algorithms [Fomin and Golovach 2014; Goyal et al. 2013, 2015; Pinter et al. 2014; Shachnai and Zehavi 2014a, 2014b; Gabizon et al. 2015; Fomin et al. 2016]. The results also have been used in the context of exact learning [Abasi et al. 2014] and linear time constructions of some d-restriction problems [Bshouty 2015]. Lokshtanov et al. [2015] obtained a deterministic algorithm for computing an ℓ-truncation of a given matrix and obtained a deterministic version of Theorem 3.8 for those matroids whose representation can be found in deterministic polynomial time. Recently Zehavi [2013] announced a further improvement for the k-PATH algorithm, which runs in time $2.597^k \cdot n^{\mathcal{O}(1)}$. It has also been brought to our attention by Cygan [2013], in a private communication, that one can obtain a single-exponential time algorithm for MINIMUM EQUIVALENT GRAPH based on the methods described in Bodlaender et al. [2015] and Cygan et al. [2011].

REFERENCES

Hasan Abasi, Nader H. Bshouty, and Hanna Mazzawi. 2014. On exact learning monotone DNF from membership queries. In *Proceedings of the 25th International Conference on Algorithmic Learning Theory (ALT) (Lecture Notes in Comput. Sci.)*, Vol. 8776. Springer, 111–124. DOI:http://dx.doi.org/10.1007/978-3-319-11662-4_9

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

Omid Amini, Fedor V. Fomin, and Saket Saurabh. 2012. Counting subgraphs via homomorphisms. *SIAM J. Discrete Math.* 26, 2 (2012), 695–717.

Jørgen Bang-Jensen and Gregory Gutin. 2009. *Digraphs* (2nd ed.). Springer-Verlag London Ltd., London. xxii+795 pages. DOI:http://dx.doi.org/10.1007/978-1-84800-998-1 Theory, algorithms and applications.

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

Andreas Björklund, Thore Husfeldt, and Sanjeev Khanna. 2004. Approximating longest directed paths and cycles. In *Proceedings of the 31st International Colloquium, Automata, Languages and Programming (ICALP'04) (Lecture Notes in Comput. Sci.)*, Vol. 3142. Springer, 222–233.

Hans L. Bodlaender. 1993. On linear time minor tests with depth-first search. *J. Algorithms* 14, 1 (1993), 1–23.

Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. 2015. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.* 243 (2015), 86–111. DOI:http://dx.doi.org/10.1016/j.ic.2014.12.008

B. Bollobás. 1965. On generalized graphs. *Acta Math. Acad. Sci. Hungar* 16 (1965), 447–452.

Nader H. Bshouty. 2015. Linear time constructions of some d-restriction problems. In *Proceedings of the 9th International Conference on Algorithms and Complexity (CIAC'15) (Lecture Notes in Comput. Sci.)*, Vol. 9079. Springer, 74–88. DOI:http://dx.doi.org/10.1007/978-3-319-18173-8_5

J. Bunch and J. Hopcroft. 1974. Triangular factorization and inversion by fast matrix multiplication. *Math. Comp.* 28, 125 (1974), 231–236.

Jianer Chen, Joachim Kneis, Songjian Lu, Daniel Mölle, Stefan Richter, Peter Rossmanith, Sing-Hoi Sze, and Fenghui Zhang. 2009. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM J. Comput.* 38, 6 (2009), 2526–2547. DOI:http://dx.doi.org/10.1137/080716475

Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. 2007. Improved algorithms for path, matching, and packing problems. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'07)*. SIAM, 298–307.

Nathann Cohen, Fedor V. Fomin, Gregory Gutin, Eun Jung Kim, Saket Saurabh, and Anders Yeo. 2010. Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem. *J. Comput. System Sci.* 76, 7 (2010), 650–662. DOI:http://dx.doi.org/10.1016/j.jcss.2010.01.001

T. H. Cormen, Ch. Leiserson, R. Rivest, and C. Stein. 2001. *Introduction to Algorithms* (2nd ed.). MIT Press, Cambridge, MA.

Marek Cygan. 2013. Private communication. (2013).

Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer.

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'11)*. IEEE, 150–159.

R. G. Downey and M. R. Fellows. 1999. *Parameterized Complexity*. Springer-Verlag, New York.

Jack Edmonds. 1967. Optimum branchings. *J. Res. Nat. Bur. Standards Sect. B* 71B (1967), 233–240.

Fedor V. Fomin, Petr Golovach, Fahad Panolan, and Saket Saurabh. 2016. Editing to connected f-degree graph. In *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS'16) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 47. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 36:1–36:14. DOI:http://dx.doi.org/10.4230/LIPIcs.STACS.2016.36

Fedor V. Fomin and Petr A. Golovach. 2014. Long circuits and large euler subgraphs. *SIAM J. Discrete Math.* 28, 2 (2014), 878–892. DOI:http://dx.doi.org/10.1137/130936816

Fedor V. Fomin and Petteri Kaski. 2013. Exact exponential algorithms. *Commun. ACM* 56, 3 (2013), 80–88. DOI:http://dx.doi.org/10.1145/2428556.2428575

Fedor V. Fomin and Dieter Kratsch. 2011. *Exact Exponential Algorithms*. Springer.

Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. 2014b. Representative sets of product families. In *Proceedings of the 22nd Annual European Symposium on Algorithms (ESA'14) (Lecture Notes in Comput. Sci.)*, Vol. 8737. Springer, 443–454. DOI:http://dx.doi.org/10.1007/978-3-662-44777-2_37

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

Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. 2014a. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*. SIAM, 142–151. DOI:http://dx.doi.org/10.1137/1.9781611973402.10

P. Frankl. 1982. An extremal problem for two families of sets. *Eur. J. Combin.* 3, 2 (1982), 125–127.

Ariel Gabizon, Daniel Lokshtanov, and Michal Pilipczuk. 2015. Fast algorithms for parameterized problems with relaxed disjointness constraints. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA'15)*, Vol. 9294. Springer, 545–556. DOI:http://dx.doi.org/10.1007/978-3-662-48350-3_46

Harold N. Gabow and Shuxin Nie. 2008. Finding a long directed cycle. *ACM Trans. Algorithms* 4, 1 (2008), Article No. 7.

Prachi Goyal, Neeldhara Misra, and Fahad Panolan. 2013. Faster deterministic algorithms for r-dimensional matching using representative sets. In *Proceedings of the 33rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'13)*, Vol. 24. 237–248. DOI:http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2013.237

Prachi Goyal, Pranabendu Misra, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. 2015. Finding even subgraphs even faster. In *Proceedings of the 35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'15)*. 434–447. DOI:http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2015.434

Harry T. Hsu. 1975. An algorithm for finding a minimal equivalent graph of a digraph. *J. Assoc. Comput. Mach.* 22 (1975), 11–16.

Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. 2008. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica* 52, 2 (2008), 114–132. DOI:http://dx.doi.org/10.1007/s00453-007-9008-7

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which problems have strongly exponential complexity. *J. Comput. System Sci.* 63, 4 (2001), 512–530.

Stasys Jukna. 2011. *Extremal Combinatorics*. Springer Verlag, Berlin.

Ton Kloks. 1994. *Treewidth, Computations and Approximations (Lecture Notes in Comput. Sci.)*, Vol. 842. Springer.

Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. 2008. Divide-and-color. In *Proceedings of the 34th International Workshop Graph-Theoretic Concepts in Computer Science (WG'08) (Lecture Notes in Comput. Sci.)*, Vol. 4271. Springer, 58–67.

Ioannis Koutis. 2008. Faster algebraic algorithms for path and packing problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP'08) (Lecture Notes in Comput. Sci.)*, Vol. 5125. 575–586.

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'09) (Lecture Notes in Comput. Sci.)*, Vol. 5555. Springer, 653–664.

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

Stefan Kratsch and Magnus Wahlström. 2012. Representative sets and irrelevant vertices: New tools for kernelization. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS'12)*. IEEE, 450–459.

Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. 2015. Deterministic truncation of linear matroids. In *Proceedings of the 42nd International Colloquium of Automata, Languages and Programming (ICALP'15) (Lecture Notes in Comput. Sci.)*, Vol. 9134. Springer, 922–934. DOI:http://dx.doi.org/10.1007/978-3-662-47672-7_75

L. Lovász. 1977. Flats in matroids and geometric graphs. In *Combinatorial Surveys (Proceedings of the 6th British Combinatorial Conference, Royal Holloway Coll., Egham)*. Academic Press, London, 45–86.

S. Martello. 1978. An algorithm for finding a minimal equivalent graph of a strongly connected digraph. *Computing* 21, 3 (1978), 183–194. DOI:http://dx.doi.org/10.1007/BF02253052

Silvano Martello and Paolo Toth. 1982. Finding a minimum equivalent graph of a digraph. *Networks* 12, 2 (1982), 89–100. DOI:http://dx.doi.org/10.1002/net.3230120202

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

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

Michael Mitzenmacher and Eli Upfal. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press.

B. 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. 239–254. DOI:http://dx.doi.org/10.1016/S0304-0208(08)73110-4

Dennis M. Moyles and Gerald L. Thompson. 1969. An algorithm for finding a minimum equivalent graph of a digraph. *J. ACM* 16, 3 (July 1969), 455–460. DOI:http://dx.doi.org/10.1145/321526.321534

Kazuo Murota. 2000. *Matrices and Matroids for Systems Analysis*. Vol. 20. Springer.

Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. 1995. Splitters and near-optimal derandomization. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*. IEEE, 182–191.

James G. Oxley. 2006. *Matroid Theory*. Vol. 3. Oxford University Press.

Christos H. Papadimitriou and Mihalis Yannakakis. 1996. On limited nondeterminism and the complexity of the V-C dimension. *J. Comput. Syst. Sci.* 53, 2 (1996), 161–170.

Ron Y. Pinter, Hadas Shachnai, and Meirav Zehavi. 2014. Deterministic parameterized algorithms for the graph motif problem. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS'14) (Lecture Notes in Comput. Sci.)*, Vol. 8635. Springer, 589–600. DOI:http://dx.doi.org/10.1007/978-3-662-44465-8_50

Jürgen Plehn and Bernd Voigt. 1991. Finding minimally weighted subgraphs. In *Proceedings of the 16th Workshop on Graph-Theoretic Concepts in Computer Science (WG'91) (Lecture Notes in Comput. Sci.)*, Vol. 484. Springer, 18–29.

Hadas Shachnai and Meirav Zehavi. 2014a. Parameterized algorithms for graph partitioning problems. In *Proceedings of the 40th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'14) (Lecture Notes in Comput. Sci.)*, Vol. 8747. Springer, 384–395. DOI:http://dx.doi.org/10.1007/978-3-319-12340-0_32

Hadas Shachnai and Meirav Zehavi. 2014b. Representative families: A unified tradeoff-based approach. In *Proceedings of the 22nd Annual European Symposium on Algorithms (ESA'14) (Lecture Notes in Comput. Sci.)*, Vol. 8737. Springer, 786–797. DOI:http://dx.doi.org/10.1007/978-3-662-44777-2_65

Zs. 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.

Zs. 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.

Ryan Williams. 2009. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.* 109, 6 (2009), 315–318.

Virginia Vassilevska Williams. 2012. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC'12)*. ACM, 887–898.

Meirav Zehavi. 2013. Mixing color coding-related techniques. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA'13) (Lecture Notes in Comput. Sci.)*, Vol. 9294. Springer, 1037–1049.

Received November 2014; revised February 2016; accepted June 2016