Representative Families: A Unified Tradeoff-Based Approach

Hadas Shachnai and Meirav Zehavi

Department of Computer Science, Technion, Haifa 32000, Israel
`{hadas,meizeh}@cs.technion.ac.il`

Abstract. Let $M = (E, \mathcal{I})$ be a matroid, and let $\mathcal{S}$ be a family of subsets of size p of E. A subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ represents $\mathcal{S}$ if for every pair of sets $X \in \mathcal{S}$ and $Y \subseteq E \setminus X$ such that $X \cup Y \in \mathcal{I}$, there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y such that $\widehat{X} \cup Y \in \mathcal{I}$. Fomin et al. (*Proc. ACM-SIAM Symposium on Discrete Algorithms, 2014*) introduced a powerful technique for fast computation of representative families for uniform matroids. In this paper, we show that this technique leads to a unified approach for substantially improving the running times of parameterized algorithms for some classic problems. This includes, among others, k-Partial Cover, k-Internal Out-Branching, and Long Directed Cycle. Our approach exploits an interesting tradeoff between running time and the size of the representative families.

1   Introduction

The theory of matroids is unique in the extent to which it connects such disparate branches of combinatorial theory and algebra as graph theory, combinatorial optimization, linear algebra, and algorithm theory. Marx [19] 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 [20].

Let E be a universe of n elements, and $\mathcal{I}$ a family of subsets of size at most k of E, for some $k \in \mathbb{N}$, i.e., $\mathcal{I} \subseteq \{S \subseteq E : |S| \leq k\}$. Then, $U_{n,k} = (E, \mathcal{I})$ is called a *uniform matroid*. Consider such a matroid and a family $\mathcal{S}$ of p-subsets of E, i.e., sets of size p. A subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ *represents* $\mathcal{S}$ if for every pair of sets $X \in \mathcal{S}$ and $Y \subseteq E \setminus X$, such that $X \cup Y \in \mathcal{I}$ (i.e., $|Y| \leq (k-p)$), there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y. In other words, if a set Y can be extended to an independent set (of size at most k) by adding a subset from $\mathcal{S}$, then it can also be extended to an independent set (of the same size) by adding a subset from $\widehat{\mathcal{S}}$.

The *Two Families Theorem* of Bollobás [2] implies that for any uniform matroid $U_{n,k} = (E, \mathcal{I})$ and a family $\mathcal{S}$ of p-subsets of E, for some $1 \leq p \leq k$, there is a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ of size $\binom{k}{p}$ that represents $\mathcal{S}$. For more general matroids, the generalization of Lovász for this theorem, given in [17], implies a similar result, and algorithms based on this generalization are given in [19] and [10].

A parameterized algorithm with parameter k has running time $O^*(f(k))$ for some function f, where O^* hides factors polynomial in the input size. A fast computation of representative families for uniform matroids plays a central role

in obtaining better running times for such algorithms. Indeed, after each stage, in which the algorithm computes a family $\mathcal{S}$ of sets that are partial solutions,[1] we compute a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ that represents $\mathcal{S}$. Then, each reference to $\mathcal{S}$ can be replaced by a reference to $\widehat{\mathcal{S}}$. The representative family $\widehat{\mathcal{S}}$ contains "enough" sets from $\mathcal{S}$; therefore, such replacement preserves the correctness of the algorithm. Thus, if we can compute fast representative families that are small enough, we can substantially improve the running time of the algorithm.

For uniform matroids, Monien [20] computed representative families of size $\sum_{i=0}^{k-p} p^i$ in time $O(|\mathcal{S}|p(k-p)\sum_{i=0}^{k-p} p^i)$, and Marx [18] computed representative families of size $\binom{k}{p}$ in time $O(|\mathcal{S}|^2 p^{k-p})$. Recently, Fomin et al. [10] introduced a powerful technique which enables to compute representative families of size $\binom{k}{p}2^{o(k)}\log n$ in time $O(|\mathcal{S}|(k/(k-p))^{k-p}2^{o(k)}\log n)$, thus significantly improving the previous results.

In this paper, we show that the technique of [10] leads to a unified tradeoff-based approach for substantially improving the running time of parameterized algorithms for some classic problems. In particular, we demonstrate the applicability of our approach, for the following problems, among others (see also Section 1.2).

k-**Partial Cover (k-PC):** Given a universe U, a family $\mathcal{S}$ of subsets of U and a parameter $k \in \mathbb{N}$, find the smallest number of sets in $\mathcal{S}$ whose union contains at least k elements.

k-**Internal Out-Branching (k-IOB):** Given a *directed* graph $G = (V, E)$ and a parameter $k \in \mathbb{N}$, decide if G has an *out-branching* (i.e., a spanning tree having exactly one node of in-degree 0) with at least k nodes of out-degree ≥ 1.

1.1   Prior Work

The k-PC problem generalizes the well-known k-DOMINATING SET (k-DS) problem, defined as follows. Given a graph $G = (V, E)$ and a parameter $k \in \mathbb{N}$, find the smallest size of a set $U \subseteq V$ such that the number of nodes that belong to U or are neighbors of nodes in U is at least k. If k-PC can be solved in time $t(|U|, |\mathcal{S}|, k)$, then k-DS can be solved in time $t(|V|, |V|, k)$ (see, e.g., [3]). Note that the special cases of k-PC and k-DS in which $k = n$, are the classical NP-complete SET COVER and DOMINATING SET problems [11], respectively. Table 1 presents a summary of known parameterized algorithms for k-PC and k-DS. We note that the parameterized complexity of k-PC and k-DS has been studied also with respect to other parameters and for more restricted inputs (see, e.g., [3, 9, 25]).

The k-IOB problem is of interest in database systems [6]. A special case of k-IOB, called k-INTERNAL SPANNING TREE (k-IST), asks if a given *undirected* graph $G = (V, E)$ has a spanning tree with at least k internal nodes. An interesting application of k-IST, for connecting cities with water pipes, is given in [23]. The k-IST problem is NP-complete, since it generalizes the HAMILTONIAN PATH problem [12]; thus, k-IOB is also NP-complete. Table 2 presents a summary of known parameterized algorithms for k-IOB and k-IST. More details on k-IOB, k-IST and variants of these problems can be found in the excellent surveys of [21, 24].

---

[1] Typically, this is done in algorithms that use dynamic programming.

Reference	Deterministic\Randomized	Variant	Running Time
Bonnet et al. [3]	*det*	k-PC	$O^*(4^k k^{2k})$
Bläser [1]	*rand*	k-PC	$O^*(5.437^k)$
Kneis et al. [15]	*det*	k-DS	$O^*((16+\epsilon)^k)$
	rand	k-DS	$O^*((4+\epsilon)^k)$
Chen et al. [4]	*det*	k-DS	$O^*(5.437^k)$
Kneis [14]	*det*	k-DS	$O^*((4+\epsilon)^k)$
Koutis et al. [16]	*rand*	k-DS	$O^*(2^k)$
This paper	**det**	**k-PC**	$\mathbf{O^*(2.619^k)}$

Table 1. Known parameterized algorithms for k-PC and k-DS.

Reference	Deterministic\Randomized	Variant	Running Time
Priesto et al. [22]	*det*	k-IST	$O^*(2^{O(k \log k)})$
Gutin al. [13]	*det*	k-IOB	$O^*(2^{O(k \log k)})$
Cohen et al. [5]	*det*	k-IOB	$O^*(55.8^k)$
	rand	k-IOB	$O^*(49.4^k)$
Fomin et al. [8]	*det*	k-IOB	$O^*(16^{k+o(k)})$
Fomin et al. [7]	*det*	k-IST	$O^*(8^k)$
Zehavi [26]	*rand*	k-IOB	$O^*(4^k)$
This paper	**det**	**k-IOB**	$\mathbf{O^*(6.855^k)}$

**Table 2.** Known parameterized algorithms for k-IOB and k-IST.

1.2   Our Results

Given a uniform matroid $U_{n,k} = (E, \mathcal{I})$ and a family $\mathcal{S}$ of p-subsets of E, we compute a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ of size $\dfrac{(ck)^k}{p^p(ck-p)^{k-p}} 2^{o(k)} \log n$ which represents $\mathcal{S}$, in time $O(|\mathcal{S}|((ck)/(ck-p))^{k-p} 2^{o(k)} \log n)$, for any fixed $c \geq 1$. Indeed, taking $c = 1$, we have the result of Fomin et al. [10]. As c grows larger, the size of $\widehat{\mathcal{S}}$ increases, with a corresponding decrease in computation time. This enables to obtain better running times for the algorithms for LONG DIRECTED CYCLE, WEIGHTED k-PATH and WEIGHTED k-TREE, as given in [10].

In particular, we use this approach to develop *deterministic* algorithms that solve k-PC and k-IOB in times $O^*(2.619^k)$ and $O^*(6.855^k)$, respectively. We thus significantly improve the *randomized* algorithm with the best known $O^*(5.437^k)$ running time for k-PC [1], and the deterministic algorithm with the best known $O^*(16^{k+o(k)})$ running time for k-IOB [8]. This also improves the running times of the best known deterministic algorithms for k-DS and k-IST (see Tables 1 and 2).

Technical Contribution: Our unified approach exploits an interesting tradeoff between running time and the size of the representative families. This tradeoff is made precise by using, along with the scheme of [10], a parameter $c \geq 1$, which enables a more careful selection of elements to the sets.

Indeed, towards computing a representative family $\widehat{\mathcal{S}}$, we seek a family $\mathcal{F} \subseteq 2^E$ that satisfies the following condition. For every pair of sets $X \in \mathcal{S}$, and $Y \subseteq E \backslash X$ such that $X \cup Y \in \mathcal{I}$, there is a set $F \in \mathcal{F}$ such that $X \subseteq F$, and $Y \cap F = \emptyset$ (see

Fig. 1). Then, we compute $\widehat{\mathcal{S}}$ by iterating over all $S \in \mathcal{S}$ and $F \in \mathcal{F}$ such that $S \subseteq F$. The time complexity of this iterative process is the dominant factor in the overall running time. Thus, we seek a small family $\mathcal{F}$, such that for any $S \in \mathcal{S}$, the expected number of sets in $\mathcal{F}$ containing S is small. In constructing each set $F \in \mathcal{F}$, we insert each element $e \in E$ to F with probability $p/(ck)$. For $c = 1$, we get the probability used in [10]. When we take a larger value for c, we need to construct a larger family $\mathcal{F}$ in order to satisfy the above condition. Yet, since elements in E are inserted to sets in $\mathcal{F}$ with a smaller probability, we get that for any $S \in \mathcal{S}$, the expected number of sets in $\mathcal{F}$ containing S is smaller.
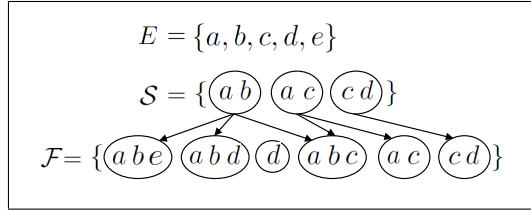
Fig. 1. An illustration of a family $\mathcal{F} \subseteq 2^E$. Assume that $n = 5, k = 4$, and $p = 2$. An arrow from a set $S \in \mathcal{S}$ to a set $F \in \mathcal{F}$ indicates that $S \subseteq F$.

Organization: Section 2 gives some definitions and notation. Section 3 presents a tradeoff between running time and the size of the representative families. Using this computation, we derive in Sections 4 and 5 our main results, which are fast parameterized algorithms for k-PC and k-IOB. Finally, Section 6 shows the improvements in running times resulting from our tradeoff-based approach for three previous applications of representative families of [10].

2   Preliminaries

We first define the weighted version of representative families.

Definition 1. *Given a matroid $U_{n,k} = (E, \mathcal{I})$, a family $\mathcal{S}$ of p-subsets of E, and a function $w : \mathcal{S} \to \mathbb{R}$, we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ max (min) represents $\mathcal{S}$ if for every pair of sets $X \in \mathcal{S}$, and $Y \subseteq E \setminus X$ such that $X \cup Y \in \mathcal{I}$, there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y such that $w(\widehat{X}) \geq w(X)$ ($w(\widehat{X}) \leq w(X)$).*

We give an illustration of a representative family in Fig. 3 (see Appendix A). The special case where $w(S) = 0$, for all $S \in \mathcal{S}$, is the unweighted version of Definition 1.

The following simple observation (used in Sections 4 and 5) asserts that representation is a transitive relation among families of subsets.

Observation 1 ([10]) *Let $U_{n,k} = (E, \mathcal{I})$ be a matroid. Let $\mathcal{S}, \mathcal{T}$ and $\mathcal{R}$ be families of p-subsets of E, and let w be a function from $\mathcal{S} \cup \mathcal{T} \cup \mathcal{R}$ to $\mathbb{R}$. If $\mathcal{S}$ max (min) represents $\mathcal{T}$ and $\mathcal{T}$ max (min) represents $\mathcal{R}$, then $\mathcal{S}$ max (min) represents $\mathcal{R}$.*

Notation: Given a set U and a nonnegative integer t, let $\binom{U}{t} = \{U' \subseteq U : |U'| = t\}$. Also, recall that an out-tree T is a directed tree having exactly one node of in-degree 0, called *the root*. We denote by V_T, E_T, $i(T)$ and $\ell(T)$ the node set, edge set, number of internal nodes (i.e., nodes of out-degree ≥ 1) and number of leaves (i.e., nodes of out-degree 0), respectively.

3 A Tradeoff-Based Approach

In this section we prove the following result.

Theorem 1. *Given a parameter $c \geq 1$, a uniform matroid $U_{n,k} = (E, \mathcal{I})$, a family $\mathcal{S}$ of p-subsets of E, and a function $w : \mathcal{S} \to \mathbb{R}$, a family $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ of size $\dfrac{(ck)^k}{p^p(ck-p)^{k-p}}2^{o(k)}\log n$ that max (min) represents $\mathcal{S}$ can be found in time $O(|\mathcal{S}|(ck/(ck-p))^{k-p}2^{o(k)}\log n + |\mathcal{S}|\log|\mathcal{S}|)$.*

Note that, in the special case where $c = 1$, we have the statement of Theorem 6 in [10]. Roughly speaking, the proof of Theorem 1 is structured as follows. We first argue that we can focus on finding a certain data structure to compute representative families. Then, we construct such a data structure that is not as efficient as required (first randomly, and then deterministically). This part contains our main contribution. Finally, we show how to improve the *"efficiency"* of this data structure (this is made precise below).

Proof. Clearly, we may assume that $|\mathcal{S}| \geq \dfrac{(ck)^k}{p^p(ck-p)^{k-p}}2^{o(k)}\log n$. Recall that our computation of representative families requires finding initially a family $\mathcal{F} \subseteq 2^E$ that satisfies the following condition. For every pair of sets $X \in \mathcal{S}$, and $Y \subseteq E\backslash X$, such that $X \cup Y \in \mathcal{I}$, there is a set $F \in \mathcal{F}$ such that $X \subseteq F$, and $Y \cap F = \emptyset$ (see Fig. 1). An (n,k,p)-*separator* is a data structure containing such a family $\mathcal{F}$, which, given a set $S \in \binom{E}{p}$, outputs the subfamily of sets in $\mathcal{F}$ that contain S, i.e., $\chi(S) = \{F \in \mathcal{F} : S \subseteq F\}$.

   To derive our fast computation, we need to construct an efficient (n,k,p)-separator, where efficiency is measured by the following parameters: $\mathcal{C} = \mathcal{C}(n,k,p)$, the number of sets in the family $\mathcal{F}$; $\tau_{\mathcal{F}} = \tau_{\mathcal{F}}(n,k,p)$, the time required to compute the family $\mathcal{F}$; $\Delta = \Delta(n,k,p)$, the maximum size of $\chi(S)$, for any $S \in \binom{E}{p}$, and $\tau_{\chi} = \tau_{\chi}(n,k,p)$, an upper bound for the time required to output $\chi(S)$, for any $S \in \binom{E}{p}$.

   Given such a separator, a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ of size $\mathcal{C}$ that max (min) represents $\mathcal{S}$ can be constructed in time $O(\tau_{\mathcal{F}} + |\mathcal{S}|\tau_{\chi} + |\mathcal{S}|\log|\mathcal{S}|)$ as follows. First, compute $\mathcal{F}$, and $\chi(S)$ for all $S \in \mathcal{S}$. Then, order $\mathcal{S} = \{S_1,...,S_{|\mathcal{S}|}\}$, such that $w(S_{i-1}) \geq w(S_i)$ ($w(S_{i-1}) \leq w(S_i)$), for all $2 \leq i \leq |\mathcal{S}|$. Finally, return all $S_i \in \mathcal{S}$ for which there is a set $F \in \mathcal{F}$ containing S_i but no S_j, for $1 \leq j < i$. Formally, return $\widehat{\mathcal{S}} = \{S_i \in \mathcal{S} : \chi(S_i)\backslash (\bigcup_{1 \leq j < i} \chi(S_j)) \neq \emptyset\}$. The correctness of this construction is proved in [10]. Thus, to prove the theorem it suffices to find an (n,k,p)-separator with parameters:

- $\mathcal{C}^* \leq \dfrac{(ck)^k}{p^p(ck-p)^{k-p}}2^{o(k)}\log n.$
- $\tau_{\mathcal{F}}^* \leq \dfrac{(ck)^k}{p^p(ck-p)^{k-p}}2^{o(k)}n\log n.$
- $\tau_{\chi}^* \leq (ck/(ck-p))^{k-p}2^{o(k)}\log n.$

We start by giving an (n,k,p)-separator, that we call Separator 1, with the following parameters, which are worse than required:

$$- \; \mathcal{C}^1 = O(\frac{(ck)^k}{p^p(ck-p)^{k-p}}k^{O(1)}\log n).$$

$$- \; \tau_{\mathcal{F}}^1 = O(\binom{2^n}{\mathcal{C}^1}n^{O(k)}).$$

$$- \; \Delta^1 = O((ck/(ck-p))^{k-p}k^{O(1)}\log n).$$

$$- \; \tau_{\chi}^1 = O(\frac{(ck)^k}{p^p(ck-p)^{k-p}}n^{O(1)}).$$

First, we give a randomized algorithm which constructs, with positive probability, an (n,k,p)-separator having the desired $\mathcal{C}^1$ and Δ^1 parameters. We then show how to deterministically construct an (n,k,p)-separator having all the desired parameter values. Let $t = \dfrac{(ck)^k}{p^p(ck-p)^{k-p}}(k+1)\ln n$, and construct the family $\mathcal{F} = \{F_1,\ldots,F_t\}$ as follows. For each $i \in \{1,\ldots,t\}$ and element $e \in E$, insert e to F_i with probability $p/(ck)$. The construction of different sets in $\mathcal{F}$, as well as the insertion of different elements into each set in $\mathcal{F}$, are independent. Clearly, $\mathcal{C}^1 = t$ is within the required bound.

For fixed sets $X \in \binom{E}{p}$, $Y \in \binom{E \setminus X}{k-p}$ and $F \in \mathcal{F}$, the probability that $X \subseteq F$ and $Y \cap F = \emptyset$ is $(\frac{p}{ck})^p(1-\frac{p}{ck})^{k-p} = \frac{p^p(ck-p)^{k-p}}{(ck)^k} = (k+1)\ln n/t$. Thus, the probability that *no* set $F \in \mathcal{F}$ satisfies $X \subseteq F$ and $Y \cap F = \emptyset$ is $(1-(k+1)\ln n/t)^t \le e^{-(k+1)\ln n} = n^{-k-1}$. There are at most n^k choices for $X \in \binom{E}{p}$ and $Y \in \binom{E \setminus X}{k-p}$; thus, applying the union bound, the probability that there exist $X \in \binom{E}{p}$ and $Y \in \binom{E \setminus X}{k-p}$, such that no set $F \in \mathcal{F}$ satisfies $X \subseteq F$ and $Y \cap F = \emptyset$, is at most $n^{-k-1} \cdot n^k = \frac{1}{n}$.

For any sets $S \in \binom{E}{p}$ and $F \in \mathcal{F}$, the probability that $S \subseteq F$ is $(p/(ck))^p$. Therefore, $|\chi(S)|$, the number of sets in $\mathcal{F}$ containing S, is a sum of t i.i.d. Bernoulli random variables with parameter $(p/(ck))^p$. Then, the expected value of $|\chi(S)|$ is $E[|\chi(S)|] = t(\frac{p}{ck})^p = (\frac{ck}{ck-p})^{k-p}(k+1)\ln n$. Applying standard Chernoff bounds, we have that the probability that $|\chi(S)| \ge 6E[|\chi(S)|]$ is upper bounded by $2^{-6E[|\chi(S)|]} \le n^{-k-1}$. There are $\binom{n}{p}$ choices for $S \in \binom{E}{p}$. Thus, by the union bound, the probability that $\Delta^1 > 6 \cdot [((ck)/(ck-p))^{k-p}(k+1)\ln n]$ is upper bounded by $\frac{1}{n}$.

So far, we have given a randomized algorithm that constructs an (n,k,p)-separator having the desired $\mathcal{C}^1$ and Δ^1 parameters with probability at least $1-\frac{2}{n} > 0$. To deterministically construct $\mathcal{F}$ in time bounded by $\tau_{\mathcal{F}}^1$, we iterate over all families of t subsets of E (there are $\binom{2^n}{\mathcal{C}^1}$ such families), where for each family $\mathcal{F}$, we test in time $n^{O(k)}$ whether Δ^1 is within the required bound, and whether for any pair of sets $X \in \binom{E}{p}$ and $Y \in \binom{E \setminus X}{k-p}$, there is a set $F \in \mathcal{F}$ such that $X \subseteq F$ and $Y \cap F = \emptyset$. Then, given a set $S \in \binom{E}{p}$, we can deterministically compute $\chi(S)$ within the stated bound for τ_{χ}^1, by iterating over $\mathcal{F}$ and inserting each set that contains S.

To obtain an (n,k,p)-separator having the parameters $\mathcal{C}^*$, $\tau_{\mathcal{F}}^*$ and τ_{χ}^*, repeatedly apply Lemmas 4.4 and 4.5 of [10] to Separator 1 (see Appendix B).   □

We note that the scheme for computing representative families, developed in the proof of Theorem 1, consists of three main stages:

1. Construct several "small" inefficient separators.
2. Given the results of Stage 1, construct an efficient (n, k, p)-separator.
3. Use the separator generated in Stage 2 to construct a representative family.

We give a pseudocode of the scheme, called RepAlg, in Appendix C.

4   An Algorithm for k-Partial Cover

We now show how to apply our scheme, RepAlg, to obtain a faster parameterized algorithm for k-PC. Let $m = |\mathcal{S}|$ be the number of sets in $\mathcal{S}$. The main idea of the algorithm is to iterate over the sets in $\mathcal{S}$ in some arbitrary order $S_1, S_2,...,S_m$, such that when we reach a set S_i, we have already computed representative families for families of "partial solutions" that include only elements from the sets $S_1,...,S_{i-1}$. Then, we try to extend the partial solutions by adding *uncovered* elements from S_i. The key observation, that leads to our improved running time, is that we cannot simply add "many" elements from S_i at once, but rather add these elements one-by-one; thus, we can compute new representative families after adding each element, which are then used in adding the next element.

The Algorithm: We now describe PCAlg, our algorithm for k-PC (see the pseudocode below). The first step solves the simple case where the solution is '1'. Then, algorithm PCAlg generates a matrix M, where each entry $M[i,j,\ell]$ holds a family that represents $Sol_{i,j,\ell}$, the family of partial solutions including j elements, obtained from a subfamily $\mathcal{S}'$ of ℓ sets in $\{S_1,..., S_i\}$, i.e., $Sol_{i,j,\ell} = \{S \subseteq (\bigcup \mathcal{S}') : \mathcal{S}' \subseteq \{S_1,..., S_i\}, |S| = j, |\mathcal{S}'| = \ell\}$.

Algorithm PCAlg iterates over all triples (i,j,ℓ), where $i \in \{1,...,m\}$, $j \in \{1,...,k\}$ and $\ell \in \{1,..., \min\{i,k\}\}$. In each iteration, corresponding to a triple (i,j,ℓ), PCAlg computes $M[i,j,\ell]$, by using $M[i-1,j',\ell-1]$, for all $1 \leq j' \leq j$, and $M[i-1,j,\ell]$. In other words, PCAlg computes a family that represents $Sol_{i,j,\ell}$ by using families that represent $Sol_{i-1,j',\ell-1}$, for all $1 \leq j' \leq j$, and $Sol_{i-1,j,\ell}$. In particular, algorithm PCAlg adds elements in S_i one-by-one to sets in $M[i-1,j',s-1]$, for all $1 \leq j' \leq j$. After adding an element, PCAlg computes (in Step 7) new representative families, to be used in adding the next element. Let $S_i = \{s_1,..., s_r\}$. Then, PCAlg computes a family $\mathcal{A}_{r'j'}$, for all $1 \leq r' \leq r$ and $0 \leq j' \leq j$, that represents the family of partial solutions including j' elements, obtained from $\{s_1,..., s_{r'}\}$ and $(\ell-1)$ sets in $\{S_1,...,S_{i-1}\}$. The family $\mathcal{A}_{r'j'}$ is computed by calling RepAlg with the *family parameter* containing the union of $\mathcal{A}_{r'-1j'}$ and the family of sets obtained by adding $s_{r'}$ to sets in $\mathcal{A}_{r'-1j'-1}$.

Suppose the solution is ℓ^*. Then, using representative families guarantees that each entry $M[i,j,\ell]$ holds "enough" sets from $Sol_{i,j,\ell}$, such that when the algorithm terminates, $M[m,k,\ell^*] \neq \emptyset$. Moreover, using representative families guarantees that each entry $M[i,j,\ell]$ does not hold "too many" sets from $Sol_{i,j,\ell}$, thereby yielding an improved running time.

Correctness and Running Time: We first state a lemma referring to Steps 5–8 in PCAlg (we give the proof in Appendix D). In this lemma, we use the

---

Algorithm 1 $\mathsf{PCAlg}(U, k, \mathcal{S} = \{S_1, \ldots, S_m\})$

1: **if** there is $S \in \mathcal{S}$ s.t. $|S| \geq k$ **then return** 1. **end if**
2: let M be a matrix that has an entry $[i, j, \ell]$ for all $0 \leq i \leq m, 1 \leq j \leq k$ and $0 \leq \ell \leq k$, initialized to $\emptyset$.
3: **for** $i = 1, \ldots, m,\ j = 1, \ldots, k,\ \ell = 1, \ldots, \min\{i, k\}$ **do**
4:     let $S_i = \{s_1, \ldots, s_r\}$.
5:     $\mathcal{A}_{0,0} \Leftarrow \{\emptyset\}$, and **for** $j' = 1, \ldots, j$ **do** $\mathcal{A}_{0,j'} \Leftarrow M[i-1, j', \ell-1]$. **end for**
6:     **for** $r' = 1, \ldots, r,\ j' = 0, \ldots j$ **do**
7:         $\mathcal{A}_{r',j'} \Leftarrow \mathsf{RepAlg}(U, k, \mathcal{A}_{r'-1,j'} \cup \{S \cup \{s_{r'}\} : j' \geq 1,\ S \in \mathcal{A}_{r'-1,j'-1},\ s_{r'} \notin S\})$.
8:     **end for**
9:     $M[i, j, \ell] \Leftarrow \mathsf{RepAlg}(U, k, M[i-1, j, \ell] \cup \mathcal{A}_{r,j})$.
10: **end for**
11: return the smallest ℓ such that $M[m, k, \ell] \neq \emptyset$.

---

following notation. For all $0 \leq i \leq m, 1 \leq j \leq k$ and $0 \leq \ell \leq k$, we let $\mathcal{A}_{i,j,\ell}^*$ denote the family of sets containing j elements, constructed by adding elements from S_i to sets in $(\bigcup_{1 \leq j' \leq j} M[i-1, j', \ell-1]) \cup \{\emptyset\}$, i.e., $\mathcal{A}_{i,j,\ell}^* = \{S \cup S_i' : S \in (\bigcup_{1 \leq j' \leq j} M[i-1, j', \ell-1]) \cup \{\emptyset\}, S_i' \subseteq S_i, |S \cup S_i'| = j\}$.

Lemma 2 *Consider an iteration of Step 3 in* PCAlg*, corresponding to some values i, j and ℓ. Then, the family $\mathcal{A}_{r,j}$ represents the family $\mathcal{A}_{i,j,\ell}^*$.*

We use Lemma 2 in proving the next lemma, which shows the correctness of PCAlg (see Appendix E).

Lemma 3 *For all $0 \leq i \leq m, 1 \leq j \leq k$ and $0 \leq \ell \leq k$, $M[i, j, \ell]$ represents $Sol_{i,j,\ell}$.*

We summarize in the next result.

Theorem 4. PCAlg *solves k-PC in time $O(2.619^k |\mathcal{S}| \log^2 |U|)$.*

Proof. Lemma 3 and Step 11 imply that PCAlg solves k-PC. Also, Lemmas 2 and 3, and the way RepAlg proceeds, imply that PCAlg runs in time

$$O(2^{o(k)} |\mathcal{S}| \log^2 |U| \cdot \max_{0 \leq t \leq k} \left\{ \frac{(ck)^k}{t^t (ck - t)^{k-t}} \left(\frac{ck}{ck - t} \right)^{k-t} \right\})$$

By choosing $c = 1.447$, the maximum is obtained at $t = \alpha k$, where $\alpha \cong 0.55277$. Thus, PCAlg runs in time $O(2.61804^k |\mathcal{S}| \log^2 |U|)$. $\square$

5 An Algorithm for k-INTERNAL OUT-BRANCHING

We show below how to use our scheme, RepAlg, to obtain a faster parameterized algorithm for k-IOB. We first define an auxiliary problem called (k,t)-TREE, which requires finding a tree on a "small" number of nodes, rather than a spanning tree. Given a directed graph $G = (V, E)$, a node $r \in V$, and nonnegative integers k and t, the (k,t)-TREE problem asks if G contains an out-tree T rooted at r, such that $i(T) = k$ and $\ell(T) = t$. The following lemma implies that we can focus on solving (k,t)-TREE.

Lemma 5 ([26]) *If (k, t)-TREE can be solved in time $\tau(G, k, t)$, then k-IOB can be solved in time $O(|V|(|E| + \sum_{1 \leq t \leq k} \tau(G, k, t)))$.*

Next, we show how to solve (k,t)-TREE. Our solution technique is based on iterating over all pairs of nodes $v, u \in V$, and all values $0 \le i \le k-1$ and $0 \le \ell \le t$. When we reach such v, u, i and ℓ, we have already computed, for all $v', u' \in V$, $0 \le i' \le i$, and $0 \le \ell' \le \ell$ satisfying $i'+\ell' < i+\ell$, representative families for families of sets that are "partial solutions". Each such set contains nodes of an out-tree of G that is rooted at v', includes u' as a leaf (unless $v'=u'$) and consists of i' internal nodes (excluding v') and ℓ' leaves (excluding u'). We then try to "connect" out-trees represented by partial solutions in a manner that results in a *legal* out-tree—i.e., an out-tree of G that is rooted at v, includes u as a leaf (unless $v=u$) and consists of i internal nodes (excluding v) and ℓ leaves (excluding u). In constructing a set of such legal out-trees, we add families of "small" partial solutions one-by-one, so we can compute new representative families after adding each family, and then use them in adding the next one—this is a crucial point in obtaining our improved running time. The construction itself is quite technical. On a high level, it consists of iterating over some trees that indicate which families of partial solutions should be currently used, and in which order they should be added.

Some Definitions: Let $d \ge 2$ be a constant (to be determined). Given nodes $v, u \in V$, $0 \le i \le k-1$ and $0 \le \ell \le t$, let $\mathcal{T}_{v,u,i,\ell}$ be the set of out-trees of G rooted at v, having exactly i internal nodes and ℓ leaves, excluding v and u, where $v=u$ or u is a leaf. Also, let $Sol_{v,u,i,\ell} = \{V_T \setminus \{v,u\} : T \in \mathcal{T}_{v,u,i,\ell}\}$. Given nodes $v, u \in V$, let $\mathcal{C}_{v,u}$ be the set of trees C rooted at v, where $v=u$ or u is a leaf, $V_C \subseteq V$, and $3 \le |V_C| \le 4d$. Given a node v of a rooted tree T, let $f_T(v)$ be the father of v in T.

The Algorithm: We now describe TreeAlg, our algorithm for (k,t)-TREE (see the pseudocode below). TreeAlg first generates a matrix M, where each entry $M[v,u,i,\ell]$ holds a family that represents $Sol_{v,u,i,\ell}$. TreeAlg iterates over all $v, u \in V$, $i \in \{0,...,k-1\}$, and $\ell \in \{0,...,t\}$ such that $1 \le i+\ell$. Next, consider some iteration, corresponding to such v, u, i and ℓ.

The goal in each iteration is to compute $M[v,u,i,\ell]$, by using entries that are already computed. Algorithm TreeAlg generates a matrix N, where each entry $N[C]$ holds a family that represents the subfamily of $Sol_{v,u,i,\ell}$ including the node set (excluding v and u) of each out-tree $T \in \mathcal{T}_{v,u,i,\ell}$ *complying* with the rooted tree C as follows (see Fig. 2): (1) for any two nodes $v', u' \in V_C$, v' is an ancestor of u' in C iff v' is an ancestor of u' in T, (2) the leaves in C are leaves in T, and (3) in the forest obtained by removing V_C from T, each tree has at most $(k+t)/d$ nodes and at most two neighbors in T from V_C. Roughly speaking, each entry $N[C]$ is easier to compute than the entry $M[v,u,i,\ell]$, since C "guides" us through the computation as follows. The rooted tree C implies which entries in M are relevant to $N[C]$, in which order they should be used, and, in particular, it ensures that these are only entries of the form $M[v',u',i',\ell']$, where $i'+\ell' \le (k+t)/d$. This bound on $i'+\ell'$ ensures that the families for which we compute representative families are "small", thereby reducing the running time of calls to RepAlg. Next, consider an iteration corresponding to some $C \in \mathcal{C}_{v,u}$.

The current goal is to compute $N[C]$ by using the guidance of C, as we now describe in detail. Algorithm TreeAlg generates a matrix L, where each entry $L[j, i', \ell']$ holds a family that represents the family of node sets, excluding nodes

in V_C, of forests in $\mathcal{F}_{v,u,C,j,i',\ell'}$, which is defined as follows. The set $\mathcal{F}_{v,u,C,j,i',\ell'}$ includes each subforest F' of G complying with the subforest F of C induced by $\{w_1,...,w_j\}$ in a manner similar to the above compliance of an out-tree $T \in \mathcal{T}_{v,u,i,\ell}$ with C, such that: (1) $V_{F'} \cap (V_C \setminus V_F) = \emptyset$, and (2) the number of internal nodes (leaves) in F', excluding nodes in V_F, is i' (ℓ'). Informally, we consider such a subforest F' as a stage towards computing an out-tree $T \in \mathcal{T}_{v,u,i,\ell}$ that complies with C. Indeed, note that $\mathcal{F}_{v,u,C,|V_C|,i^*,\ell^*}$ is the set of out-trees in $\mathcal{T}_{v,u,i,\ell}$ that comply with C. Roughly speaking, the matrix L is computed by using dynamic programming and algorithm RepAlg (in Steps 8–12) as follows. Each entry in L is computed by adding node sets of certain trees to node sets of forests computed at a previous stage, and then calling algorithm RepAlg to compute a representative family for the result.
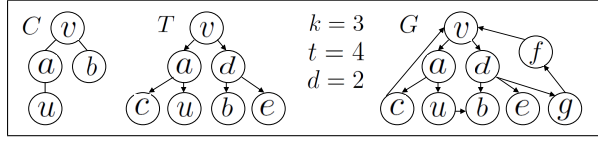
Fig. 2. An example for an out-tree $T \in \mathcal{T}_{v,u,2,3}$, complying with the rooted tree $C \in \mathcal{C}_{v,u}$.

Algorithm 2 TreeAlg$(G=(V,E), r, k, t)$

1: let M be a matrix that has an entry $[v,u,i,\ell]$ for all $v,u \in V$, $0 \le i \le k-1$ and $0 \le \ell \le t$, which is initialized to $\emptyset$.
2: $M[v,u,0,0] \Leftarrow \{\emptyset\}$ for all $v,u \in V$ s.t. $(v,u) \in E$ or $v=u$.
3: **for** all $v,u \in V$, $i=0,\ldots,k-1$, $\ell=0,\ldots,t$ s.t. $1 \le i+\ell$ **do**
4:    let N be a matrix that has an entry $[C]$ for all $C \in \mathcal{C}_{v,u}$.
5:    **for** all $C \in \mathcal{C}_{v,u}$ **do**
6:       let $V_C = \{w_1,...,w_{|V_C|}\}$ where $w_1 = v$, $i^* = i+1-i(C)$, and $\ell^* = \ell+|\{u\}\setminus\{v\}|-\ell(C)$.
7:       let L be a matrix that has an entry $[j,i',\ell']$ for all $1 \le j \le |V_C|$, $0 \le i' \le i^*$ and $0 \le \ell' \le \ell^*$, which is initialized to $\emptyset$.
8:       $L[1,i',\ell'] \Leftarrow \{U \in M[v,v,i',\ell'] : U \cap V_C = \emptyset\}$ for all $0 \le i' \le i^*$ and $0 \le \ell' \le \ell^*$ s.t. $i'+\ell' \le (k+t)/d$.
9:       **for** $j=2,\ldots,|V_C|$, $i'=0,\ldots,i^*$, $\ell'=0,\ldots,\ell^*$ **do**
10:          let $\mathcal{A}$ be the family of all sets $U \cup W$ such that $U \cap (W \cup V_C) = \emptyset$, and there are $0 \le i'' \le i'$ and $0 \le \ell'' \le \ell'$ satisfying $i'' + \ell'' \le \dfrac{k+t}{d}$ for which
            (1) $U \in M[f_C(w_j), w_j, i'', \ell'']$ and $W \in L[j-1, i'-i'', \ell'-\ell'']$; or
            (2) w_j is not a leaf in C, $\ell'' \ge 1, U \in M[w_j, w_j, i'', \ell'']$ and $W \in L[j, i'-i'', \ell'-\ell'']$.
11:          $L[j,i',\ell'] \Leftarrow$ RepAlg$(V, k+t, \mathcal{A})$.
12:       **end for**
13:       $N[C] \Leftarrow \{U \cup (V_C \setminus \{v,u\}) : U \in L[|V_C|, i^*, \ell^*]\}$.
14:    **end for**
15:    $M[v,u,i,\ell] \Leftarrow$ RepAlg$(V, k+t, \bigcup_{C \in \mathcal{C}_{v,u}} N[C])$.
16: **end for**
17: accept iff $M[r,r,k-1,t] \ne \emptyset$.

Correctness and Running Time: The following lemma implies the correctness of TreeAlg (the proof is given in Appendix F).

Lemma 6 *For all $v, u \in V$, $0 \leq i < k$ and $0 \leq \ell \leq t$, $M[v,u,i,\ell]$ represents $Sol_{v,u,i,\ell}$.*

We summarize in the next result.

Lemma 7 TreeAlg *solves (k,t)-*TREE *in time $O(2.61804^{k+t}|V|^{O(1)})$.*

Proof. Let $q = k+t$, and $0 < \epsilon < 1$ be some constant. Choose some constant $d \geq 2$ satisfying $\binom{cq}{q/d} = O(2^{\epsilon q})$ and $\frac{1}{d} \leq \epsilon$.

Lemma 6 and Step 17 imply that TreeAlg solves (k,t)-TREE. It is easy to verify that for any $0 \leq r^* \leq q$ and call RepAlg$(V, k+t, \mathcal{S})$ executed by TreeAlg, where $\mathcal{S}$ is a family of r^*-subsets of V, there is $0 \leq r' \leq \min\{r^*, q/d\}$ such that $|\mathcal{S}| \leq 2^{o(q)}|V|^{O(d)}(\frac{(cq)^q}{(r^*-r')^{r^*-r'}(cq-(r^*-r'))^{q-(r^*-r')}})(\frac{(cq)^q}{r'^{r'}(cq-r')^{q-r'}})$. Thus, the way RepAlg proceeds implies that TreeAlg runs in time

$$O\left(2^{o(q)}|V|^{O(d)}\max_{r=0}^{q}\max_{r'=0}^{\min\{q-r,q/d\}}\left\{\left(\frac{(cq)^q}{r^r(cq-r)^{q-r}}\right)\left(\frac{(cq)^q}{r'^{r'}(cq-r')^{q-r'}}\right)\left(\frac{cq}{cq-(r+r')}\right)^{q-(r+r')}\right\}\right)$$

$$= O\left(2^{o(q)}|V|^{O(1)}\max_{r=0}^{q}\left\{\left(\frac{(cq)^q}{r^r(cq-r)^{q-r}}\right)\binom{cq}{q/d}\left(\frac{(c+1/d)q}{cq-r}\right)^{q-r}\right\}\right)$$

$$= O\left(2^{\epsilon q+o(q)}|V|^{O(1)}\max_{r=0}^{q}\left\{\left(\frac{(cq)^q}{r^r(cq-r)^{q-r}}\right)\left(\frac{(c+\epsilon)q}{cq-r}\right)^{q-r}\right\}\right).$$

By choosing $c = 1.447$ and a small enough $\epsilon > 0$, the maximum is obtained at $r = \alpha k$, where $\alpha \cong 0.55277$. Thus, TreeAlg runs in time $O(2.61804^{k+t}|V|^{O(1)})$.   □

Lemmas 5 and 7 imply the following theorem.

Theorem 8. *k-*IOB *can be solved in time $O(6.85414^k|V|^{O(1)})$.*

6   Improving Known Applications

Fomin et al. [10] proved that LONG DIRECTED CYCLE, WEIGHTED k-PATH and WEIGHTED k-TREE can be solved in times $O(8^k|E|\log^2|V|)$, $O(2.851^k|V|\log^2|V|)$ and $O(2.851^k|V|^{O(1)})$, respectively. By replacing their computation of representative families with our scheme, RepAlg, we obtain for these problems exact algorithms with improved running times of $O(6.75^k|E|\log^2|V|)$, $O(2.619^k|V|\log^2|V|)$ and $O(2.619^k|V|^{O(1)})$, respectively. We give the details in Appendix G.

References

1. Bläser, M.: Computing small partial coverings. Inf. Proc. Let. 85(6), 327–331 (2003)
2. Bollobás, B.: On generalized graphs. Acta Math. Aca. Sci. Hun. 16, 447–452 (1965)
3. Bonnet, E., Paschos, V.T., Sikora, F.: Multiparameterizations for max k-set cover and related satisfiability problems. CoRR abs/1309.4718 (2013)
4. Chen, S., Chen, Z.: Faster deterministic algorithms for packing, matching and t-dominating set problems. CoRR abs/1306.3602 (2013)

5. Cohen, N., Fomin, F.V., Gutin, G., Kim, E.J., Saurabh, S., Yeo, A.: Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem. J. Comput. Syst. Sci. 76(7), 650–662 (2010)

6. Demers, A., Downing, A.: Minimum leaf spanning tree. US Patent no. 6,105,018 (August 2013)

7. Fomin, F.V., Gaspers, S., Saurabh, S., Thomassé, S.: A linear vertex kernel for maximum internal spanning tree. J. Comput. Syst. Sci. 79(1), 1–6 (2013)

8. Fomin, F.V., Grandoni, F., Lokshtanov, D., Saurabh, S.: Sharp separation and applications to exact and parameterized algorithms. Algorithmica 63(3), 692–706 (2012)

9. Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Subexponential algorithms for partial cover problems. Inf. Proc. Let. 111(16), 814–818 (2011)

10. Fomin, F.V., Lokshtanov, D., Saurabh, S.: Efficient computation of representative sets with applications in parameterized and exact agorithms. In: SODA (see also: CoRR abs/1304.4626). pp. 142–151 (2014)

11. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman, New York (1979)

12. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete problems. In: STOC. pp. 47–63 (1974)

13. Gutin, G., Razgon, I., Kim, E.J.: Minimum leaf out-branching and related problems. Theor. Comput. Sci. 410(45), 4571–4579 (2009)

14. Kneis, J.: Intuitive algorithms. RWTH Aachen University pp. 1–167 (2009)

15. Kneis, J., Mölle, D., Rossmanith, P.: Partial vs. complete domination: t-dominating set. In: SOFSEM. pp. 367–376 (2007)

16. Koutis, I., Williams, R.: Limits and applications of group algebras for parameterized problems. In: ICALP. pp. 653–664 (2009)

17. Lovász, L.: Flats in matroids and geometric graphs. In: BCC. pp. 45–86 (1977)

18. Marx, D.: Parameterized coloring problems on chordal graphs. Theor. Comput. Sci. 351, 407424 (2006)

19. Marx, D.: A parameterized view on matroid optimization problems. Theor. Comput. Sci. 410, 4471–4479 (2009)

20. Monien, B.: How to find long paths efficiently. Annals Disc. Math. 25, 239–254 (1985)

21. Ozeki, K., Yamashita, T.: Spanning trees: A survey. Graphs and Combinatorics 27(1), 1–26 (2011)

22. Prieto, E., Sloper, C.: Reducing to independent set structure – the case of k-internal spanning tree. Nord. J. Comput. 12(3), 308–318 (2005)

23. Raible, D., Fernau, H., Gaspers, D., Liedloff, M.: Exact and parameterized algorithms for max internal spanning tree. Algorithmica 65(1), 95–128 (2013)

24. Salamon, G.: A survey on algorithms for the maximum internal spanning tree and related problems. Electronic Notes in Disc. Math. 36, 1209–1216 (2010)

25. Skowron, P., Faliszewski, P.: Approximating the MaxCover problem with bounded frequencies in FPT time. CoRR abs/1309.4405 (2013)

26. Zehavi, M.: Algorithms for k-internal out-branching. In: IPEC. pp. 361–373 (2013)

A   An Illustration of a Representative Family

$$E = \{a, b, c, d, e\}$$

$$\mathcal{S} = \{\,\boxed{a\ b}\ \ \boxed{a\ c}\ \ \boxed{b\ d}\ \ \boxed{b\ e}\ \ \boxed{c\ d}\ \ \boxed{c\ e}\,\}$$

$$\widehat{\mathcal{S}} = \{\,\boxed{a\ c}\ \ \boxed{b\ d}\ \ \boxed{b\ e}\ \ \boxed{c\ e}\,\}$$

Fig. 3. An illustration of a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ that represents $\mathcal{S}$ in a matroid $U_{5,4}$, with $p=2$.

Figure 3 illustrates a representative family corresponding to the matroid $U_{5,4} = \{E, \mathcal{I}\}$, where $\mathcal{I} = \{X \subseteq E : |X| \leq 4\}$. The subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ represents $\mathcal{S}$, since for every pair of sets $X \in \mathcal{S}$, and $Y \subseteq E \setminus X$ such that $|Y| \leq (k-p) = 2$, there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y. For example, the set $X = \{a,b\} \in \mathcal{S}$ is disjoint from $\{c,d\}, \{c,e\}$, and $\{d,e\}$. Indeed, the subfamily $\widehat{\mathcal{S}}$ contains the set $\{b,e\}$ that is disjoint from $\{c,d\}$, the set $\{b,d\}$ that is disjoint from $\{c,e\}$, and the set $\{a,c\}$ that is disjoint from $\{d,e\}$.

B   Proof of Theorem 1 (Continued)

In the rest of the proof, we use the following notation. Given nonnegative integers p, s and t, let $\mathcal{Z}^p_{s,t}$ denote the set of tuples $(p_1, p_2, \ldots, p_t)$ of nonnegative integers, each of value at most s, whose sum is p. Note that $|\mathcal{Z}^p_{s,t}| \leq \binom{p+t-1}{t-1} \leq 2^{t\log(p+t)}$.

Recall that our goal is to decrease the $\tau^1_{\mathcal{F}}$ and τ^1_{χ} parameters of Separator 1 to be within the required bounds for $\tau^*_{\mathcal{F}}$ and τ^*_{χ}. We achieve this goal by repeatedly applying the following two lemmas:

Lemma 9 ([10]) *If there is an (n,k,p)-separator with parameters $\mathcal{C}(n,k,p), \tau_{\mathcal{F}}(n, k,p), \Delta(n,k,p)$ and $\tau_{\chi}(n,k,p)$, then there is an (n,k,p)-separator with parameters:*

- $\mathcal{C}'(n,k,p) \leq \mathcal{C}(k^2,k,p)k^{O(1)}\log n$.
- $\tau'_{\mathcal{F}}(n,k,p) = O(\tau_{\mathcal{F}}(k^2,k,p) + \mathcal{C}(k^2,k,p)k^{O(1)}n\log n)$.
- $\Delta'(n,k,p) \leq \Delta(k^2,k,p)k^{O(1)}\log n$.
- $\tau'_{\chi}(n,k,p) = O(\tau_{\chi}(k^2,k,p) + \Delta(k^2,k,p)k^{O(1)}\log n)$.

Lemma 10 ([10]) *Let $s = \lfloor \log^2 k \rfloor$ and $t = \lceil \frac{k}{s} \rceil$. If there is an (n,k,p)-separator with parameters $\mathcal{C}(n,k,p), \tau_{\mathcal{F}}(n,k,p), \Delta(n,k,p)$ and $\tau_{\chi}(n,k,p)$, then there is an (n,k,p)-separator with parameters:*

- $\mathcal{C}'(n,k,p) \leq 2^{O(t\log n)} \cdot \displaystyle\sum_{(p_1,\ldots,p_t)\in\mathcal{Z}^p_{s,t}} \prod_{i\leq t} \mathcal{C}(n,s,p_i)$.
- $\tau'_{\mathcal{F}}(n,k,p) = O(\displaystyle\sum_{\widehat{p}\leq s} \tau_{\mathcal{F}}(n,s,\widehat{p}) + \mathcal{C}'(n,k,p)n^{O(1)})$.
- $\Delta'(n,k,p) \leq 2^{O(t\log n)} \cdot \displaystyle\max_{(p_1,\ldots,p_t)\in\mathcal{Z}^p_{s,t}} \prod_{i\leq t} \Delta(n,s,p_i)$.

$$- \tau'_\chi(n,k,p) = O(\Delta'(n,k,p) \cdot n^{O(1)} + 2^{O(t \log n)} \cdot \sum_{\widehat{p} \leq s} \tau_\chi(n,s,\widehat{p})).$$

Applying Lemma 9 to Separator 1, we get a separator with parameters:

$$- \mathcal{C}^2(n,k,p) = O(\frac{(ck)^k}{p^p(ck-p)^{k-p}} k^{O(1)} \log n) = O(\binom{ck}{p}(\frac{ck-p}{ck})^{(c-1)k} k^{O(1)} \log n).$$

$$- \tau^2_\mathcal{F}(n,k,p) = 2^{k^{O(k)}} + \frac{(ck)^k}{p^p(ck-p)^{k-p}} k^{O(1)} n \log n.$$

$$- \Delta^2(n,k,p) = O((\frac{ck}{ck-p})^{k-p} k^{O(1)} \log n).$$

$$- \tau^2_\chi(n,k,p) = k^{O(k)} \log n.$$

We now apply Lemma 10 to this separator. Recall that in Lemma 10, we set $s = \lfloor \log^2 k \rfloor$ and $t = \lceil \frac{k}{s} \rceil$. This yields a separator with parameters:

$$
\begin{aligned}
\mathcal{C}^3(n,k,p) &\leq 2^{O(t \log n)} \cdot \sum_{(p_1,\dots,p_t) \in \mathscr{Z}^p_{s,t}} \prod_{i \leq t} \mathcal{C}^2(n,s,p_i) \\
&\leq 2^{O(t \log n)} \max_{(p_1,\dots,p_t) \in \mathscr{Z}^p_{s,t}} \prod_{i \leq t} [\binom{cs}{p_i}(\frac{cs-p_i}{cs})^{(c-1)s} s^{O(1)} \log n] \\
&\leq 2^{O(t \log n)} \cdot \binom{ck}{p} \cdot \max_{(p_1,\dots,p_t) \in \mathscr{Z}^p_{s,t}} \prod_{i \leq t} (1 - \frac{p_i}{cs})^{(c-1)s} \\
&\leq 2^{O(t \log n)} \cdot \binom{ck}{p}(1 - \frac{p}{ck})^{(c-1)k}
\end{aligned}
$$

$$
\begin{aligned}
\tau^3_\mathcal{F}(n,k,p) &= O(\sum_{\widehat{p} \leq s} \tau^2_\mathcal{F}(n,s,\widehat{p}) + \mathcal{C}^3(n,k,p) n^{O(1)}) \\
&\leq 2^{s^{O(s)}} + \max_{\widehat{p} \leq s}(\frac{(cs)^s}{\widehat{p}^{\widehat{p}}(cs-\widehat{p})^{s-\widehat{p}}}) s^{O(1)} n \log n + 2^{O(t \log n)} \binom{ck}{p}(1 - \frac{p}{ck})^{(c-1)k} \\
&\leq 2^{(\log k)^{O(\log^2 k)}} + 2^{O(t \log n)} \cdot \binom{ck}{p}(1 - \frac{p}{ck})^{(c-1)k}
\end{aligned}
$$

$$
\begin{aligned}
\Delta^3(n,k,p) &\leq 2^{O(t \log n)} \cdot \max_{(p_1,\dots,p_t) \in \mathscr{Z}^p_{s,t}} \prod_{i \leq t} \Delta^2(n,s,p_i) \\
&\leq 2^{O(t \log n)} \cdot \max_{(p_1,\dots,p_t) \in \mathscr{Z}^p_{s,t}} \prod_{i \leq t} [(\frac{cs}{cs-p_i})^{s-p_i} s^{O(1)} \log n] \\
&\leq 2^{O(t \log n)} \cdot \max_{(q_1,\dots,q_t) \in \mathscr{Z}^{k-p}_{s,t}} \prod_{i \leq t} (\frac{cs}{(c-1)s+q_i})^{q_i} \\
&\leq 2^{O(t \log n)} \cdot (ck)^{k-p} \cdot \max_{(q_1,\dots,q_t) \in \mathscr{Z}^{k-p}_{s,t}} \prod_{i \leq t} (\frac{1}{((c-1)s+q_i)t})^{q_i} = (*)
\end{aligned}
$$

By Gibbs' inequality, we have that

$$(*) \leq 2^{O(t \log n)} \cdot (ck)^{k-p} \cdot \frac{1}{((c-1)st + (k-p))^{k-p}} \leq 2^{O(t \log n)} \cdot (\frac{ck}{ck-p})^{k-p}$$

$$\tau_\chi^3(n,k,p) = O(2^{O(t\log n)}(\frac{ck}{ck-p})^{k-p}n^{O(1)} + 2^{O(t\log n)}\sum_{\widehat{p}\le s}\tau_\chi^2(n,s,\widehat{p}))$$

$$\le 2^{O(t\log n)}\cdot(\frac{ck}{ck-p})^{k-p} + 2^{O(t\log n)}s^{O(s)}\log n$$

$$\le 2^{O(t\log n)}\cdot(\frac{ck}{ck-p})^{k-p}$$

Applying Lemma 9 to this separator, we get a separator with parameters:

- $\mathcal{C}^4(n,k,p) \le 2^{O(\frac{k}{\log k})}\cdot\binom{ck}{p}(1-\frac{p}{ck})^{(c-1)k}\log n.$

- $\tau_\mathcal{F}^4(n,k,p) = O(2^{(\log k)^{O(\log^2 k)}} + 2^{O(\frac{k}{\log k})}\cdot\binom{ck}{p}(1-\frac{p}{ck})^{(c-1)k}n\log n).$

- $\Delta^4(n,k,p) \le 2^{O(\frac{k}{\log k})}\cdot(\frac{ck}{ck-p})^{k-p}\log n.$

- $\tau_\chi^4(n,k,p) = O(2^{O(\frac{k}{\log k})}\cdot(\frac{ck}{ck-p})^{k-p}\log n).$

We next apply Lemma 10 again. As in the analysis of the third separator, we set $s = \lfloor\log^2 k\rfloor$ and $t = \lceil\frac{k}{s}\rceil$. This yields a separator with parameters:

$$\mathcal{C}^5(n,k,p) \le 2^{O(t\log n)}\cdot\sum_{(p_1,\dots,p_t)\in\mathcal{Z}_{s,t}^p}\prod_{i\le t}\mathcal{C}^4(n,s,p_i)$$

$$\le 2^{O(t\log n)}\max_{(p_1,\dots,p_t)\in\mathcal{Z}_{s,t}^p}\prod_{i\le t}[2^{O(\frac{s}{\log s})}\cdot\binom{cs}{p_i}(1-\frac{p_i}{cs})^{(c-1)s}\log n]$$

$$\le 2^{O(t\log n+\frac{k}{\log\log k})}\binom{ck}{p}\max_{(p_1,\dots,p_t)\in\mathcal{Z}_{s,t}^p}[\prod_{i\le t}(1-\frac{p_i}{cs})]^{(c-1)s}$$

$$\le 2^{O(t\log n+\frac{k}{\log\log k})}\binom{ck}{p}(1-\frac{p}{ck})^{(c-1)k}$$

$$\le 2^{O(t\log n+\frac{k}{\log\log k})}\frac{(ck)^k}{p^p(ck-p)^{k-p}}$$

$$\tau_\mathcal{F}^5(n,k,p) = O(\sum_{\widehat{p}\le s}\tau_\mathcal{F}^4(n,s,\widehat{p}) + \mathcal{C}^5(n,k,p)n^{O(1)})$$

$$\le s2^{(\log s)^{O(\log^2 s)}} + \max_{\widehat{p}\le s}(2^{O(\frac{s}{\log s})}\cdot\binom{cs}{\widehat{p}}(1-\frac{\widehat{p}}{cs})^{(c-1)s}n\log n)$$

$$+2^{O(t\log n+\frac{k}{\log\log k})}\frac{(ck)^k}{p^p(ck-p)^{k-p}}$$

$$\le (\log^2 k)2^{(\log\log k)^{O(\log^2\log k)}} + 2^{O(t\log n+\frac{k}{\log\log k})}\frac{(ck)^k}{p^p(ck-p)^{k-p}}$$

$$\le 2^{O(t\log n+\frac{k}{\log\log k})}\frac{(ck)^k}{p^p(ck-p)^{k-p}}$$

In the following analysis of $\Delta^5(n,k,p)$, the last transition is achieved as in the analysis of $\Delta^3(n,k,p)$.

$$\Delta^5(n,k,p) \leq 2^{O(t \log n)} \cdot \max_{(p_1,\dots,p_t) \in \mathcal{Z}_{s,t}^p} \prod_{i \leq t} \Delta^4(n,s,p_i)$$

$$\leq 2^{O(t \log n)} \max_{(p_1,\dots,p_t) \in \mathcal{Z}_{s,t}^p} \prod_{i \leq t} (2^{O(\frac{s}{\log s})} \cdot (\frac{cs}{cs-p_i})^{s-p_i} \log n)$$

$$\leq 2^{O(t \log n + \frac{k}{\log \log k})} \max_{(p_1,\dots,p_t) \in \mathcal{Z}_{s,t}^p} \prod_{i \leq t} (\frac{cs}{cs-p_i})^{s-p_i}$$

$$\leq 2^{O(t \log n + \frac{k}{\log \log k})} (\frac{ck}{ck-p})^{k-p}$$

$$\tau_\chi^5(n,k,p) = 2^{O(t \log n + \frac{k}{\log \log k})}(\frac{ck}{ck-p})^{k-p} n^{O(1)} + 2^{O(t \log n)} \sum_{\widehat{p} \leq s} \tau_\chi^4(n,s,\widehat{p})$$

$$\leq 2^{O(t \log n + \frac{k}{\log \log k})}(\frac{ck}{ck-p})^{k-p} + 2^{O(t \log n)} s^{O(s)} \log n$$

$$\leq 2^{O(t \log n + \frac{k}{\log \log k})}(\frac{ck}{ck-p})^{k-p}$$

Applying Lemma 9 to this separator, we get a separator with parameters:

- $\mathcal{C}^6(n,k,p) \leq 2^{O(\frac{k}{\log \log k})} \dfrac{(ck)^k}{p^p(ck-p)^{k-p}} \log n.$

- $\tau_\mathcal{F}^6(n,k,p) \leq 2^{O(\frac{k}{\log \log k})} \dfrac{(ck)^k}{p^p(ck-p)^{k-p}} n \log n.$

- $\tau_\chi^6(n,k,p) \leq 2^{O(\frac{k}{\log \log k})} (\dfrac{ck}{ck-p})^{k-p} \log n.$

This separator has the desired parameters $\mathcal{C}^*(n,k,p)$, $\tau_\mathcal{F}^*(n,k,p)$ and $\tau_\chi^*(n,k,p)$, and we thus conclude the proof of the theorem. $\qquad\square$

C Pseudocode for the Computation Scheme RepAlg

We give below the pseudocode for RepAlg, the computation scheme developed in Section 3.

D Proof of Lemma 2

In this section we prove the following lemma.

Lemma 11 *Consider an iteration of Step 3 in* PCAlg, *corresponding to some values i, j and ℓ. Let $\mathcal{A} = (\bigcup_{1 \leq j' \leq j} M[i-1, j', \ell-1]) \cup \{\emptyset\}$. Then, for all $0 \leq r' \leq r$ and $0 \leq j' \leq j$, $\mathcal{A}_{r',j'}$ represents $\mathcal{A}_{r',j'}^* = \{S \cup S_i' : S \in \mathcal{A}, S_i' \subseteq \{s_1,\dots,s_{r'}\}, |S \cup S_i'| = j'\}$.*

Note that Lemma 2 is a special case of Lemma 11 when $r' = r$, and $j' = j$.

Proof. By Step 5, the claim holds for $r' = 0$ and all $0 \leq j' \leq j$. Next consider some $1 \leq r' \leq r$, and assume that the claim holds for all $0 \leq r'' < r'$ and $0 \leq j'' \leq j$. By Step 7 and Observation 1, it is enough to prove that $\mathcal{B} = \{S \cup \{s_{r'}\} : j' \geq 1, S \in \mathcal{A}_{r'-1,j'-1}, s_{r'} \notin S\} \cup \mathcal{A}_{r'-1,j'}$ represents $\mathcal{A}_{r',j'}^*$. First, we get that $\mathcal{B} \subseteq \mathcal{A}_{r',j'}^*$ as follows:

Algorithm 3 RepAlg($E, k, \mathcal{S}, w$)

1: let $n' = \log^4 \log^2 k$, and $k' = \log^2 \log^2 k$.
2: **for** $p' = 0, \ldots, \min\{k', p\}$ **do**
3:     construct an (n', k', p')-separator with parameters $\mathcal{C}^1(n', k', p')$, $\tau_{\mathcal{F}}^1(n', k', p')$, $\Delta^1(n', k', p')$, and $\tau_{\chi}^1(n', k', p')$.
4: **end for**
5: given the results of Step 2, repeatedly apply Lemmas 9 and 10 to construct an (n, p, k)-separator with parameters $\mathcal{C}^*(n, k, p)$, $\tau_{\mathcal{F}}^*(n, k, p)$, and $\tau_{\chi}^*(n, k, p)$.
6: given the result of Step 5, compute the corresponding family $\mathcal{F}$ and subfamilies $\chi(S)$, for all $S \in \mathcal{S}$.
7: **for all** $F \in \mathcal{F}$ **do**
8:     let $z_F \in \{0,1\}$ be an indicator variable for using F, initialized to 0.
9: **end for**
10: order $\mathcal{S} = \{S_1, \ldots, S_{|\mathcal{S}|}\}$, such that $w(S_{i-1}) \geq w(S_i)$ $(w(S_{i-1}) \leq w(S_i))$, for all $2 \leq i \leq |\mathcal{S}|$.
11: initialize $\widehat{\mathcal{S}} \Leftarrow \emptyset$.
12: **for** $i = 1, \ldots, |\mathcal{S}|$ **do**
13:     **for all** $F \in \chi(S_i)$ such that $z_F = 0$ **do** $\widehat{\mathcal{S}} \Leftarrow \widehat{\mathcal{S}} \cup \{S_i\}$, and $z_F \Leftarrow 1$. **end for**
14: **end for**
15: return $\widehat{\mathcal{S}}$.

$$\mathcal{B} = \{S \cup \{s_{r'}\} : j' \geq 1, S \in \mathcal{A}_{r'-1, j'-1}, s_{r'} \notin S\} \cup \mathcal{A}_{r'-1, j'}$$
$$\subseteq^{(1)} \{S \cup \{s_{r'}\} : j' \geq 1, S \in \mathcal{A}_{r'-1, j'-1}^*, s_{r'} \notin S\} \cup \mathcal{A}_{r'-1, j'}^*$$
$$=^{(2)} \{S \cup S' : S \in \mathcal{A}, S' \subseteq \{s_1, \ldots, s_{r'}\}, |S \cup S'| = j'\} = \mathcal{A}_{r', j'}^*.$$

(1) by the induction hypothesis.
(2) by the definitions of $\mathcal{A}_{r'-1, j'-1}^*, \mathcal{A}_{r'-1, j'}^*$.

Now, consider some $X \in \mathcal{A}_{r', j'}^*$ and $Y \subseteq (U \setminus X)$ such that $|Y| \leq k - j'$. Since $X \in \mathcal{A}_{r', j'}^*$, either $(X \setminus \{s_{r'}\}) \in \mathcal{A}_{r'-1, j'-1}^*$ or $X \in \mathcal{A}_{r'-1, j'}^*$. In the first case, by the induction hypothesis, there is $\widehat{X} \in \mathcal{A}_{r'-1, j'-1}$ such that $\widehat{X} \cap (Y \cup \{s_{r'}\}) = \emptyset$. Then, $(\widehat{X} \cup \{s_{r'}\}) \in \mathcal{A}_{r', j'}$, and $(\widehat{X} \cup \{s_{r'}\}) \cap Y = \emptyset$. In the second case, by the induction hypothesis, there is $\widehat{X} \in \mathcal{A}_{r'-1, j'} \subseteq \mathcal{A}_{r', j'}$ such that $\widehat{X} \cap Y = \emptyset$. Thus, $\mathcal{B}$ indeed represents $\mathcal{A}_{r', j'}^*$. $\square$

E   Proof of Lemma 3

By Step 2, the claim holds for all $1 \leq j \leq k$ and $0 \leq \ell \leq k$ when $i = 0$, and for all $0 \leq i \leq m$ and $1 \leq j \leq k$ when $\ell = 0$, or $i < \ell \leq k$. Next, consider some $1 \leq i \leq m, 1 \leq j \leq k$ and $1 \leq \ell \leq \min\{i, k\}$, and assume that the claim holds for $i - 1$, and all $1 \leq j' \leq j$ and $0 \leq \ell' \leq \ell$.

Let $Sol_{i,j,\ell}^*$ be the family of partial solutions including j elements, obtained from a subfamily $\mathcal{S}'$ that includes S_i and $(\ell - 1)$ sets in $\{S_1, \ldots, S_{i-1}\}$, i.e., $Sol_{i,j,\ell}^* = \{S \subseteq (\bigcup \mathcal{S}') : \{S_i\} \subseteq \mathcal{S}' \subseteq \{S_1, \ldots, S_i\}, |S| = j, |\mathcal{S}'| = \ell\}$. We now show that $\mathcal{A}_{i,j,\ell}^*$ represents $Sol_{i,j,\ell}^*$, which, by Lemma 2 and Observation 1, implies that $\mathcal{A}_{r,j}$ represents $Sol_{i,j,\ell}^*$. First, the induction hypothesis implies that $(\bigcup_{1 \leq j' \leq j} \text{M}[i -$

$1, j', \ell - 1]) \subseteq (\bigcup_{1 \leq j' \leq j} Sol_{i-1,j',\ell-1})$, and thus $\mathcal{A}^*_{i,j,\ell} \subseteq Sol^*_{i,j,\ell}$. Second, consider a pair of sets $X \in Sol^*_{i,j,\ell}$, and $Y \subseteq (U \setminus X)$ such that $|Y| \leq k - j$. Since $X \in Sol^*_{i,j,\ell}$, there are $X' \in (\bigcup^j_{j'=1} Sol_{i-1,j',\ell-1}) \cup \{\emptyset\}$, and $S \subseteq S_i$ such that $X = X' \cup S$. If $X' = \emptyset$, then $X \in \mathcal{A}^*_{i,j,\ell}$. Else, by the induction hypothesis, there is $\widehat{X}' \in$ M$[i-1,|X'|,\ell-1]$ such that $\widehat{X}' \cap (Y \cup (S \setminus X')) = \emptyset$. We get that $(\widehat{X}' \cup (S \setminus X')) \in \mathcal{A}^*_{i,j,\ell}$, and $(\widehat{X}' \cup (S \setminus X')) \cap Y = \emptyset$. Thus, $\mathcal{A}^*_{i,j,\ell}$ represents $Sol^*_{i,j,\ell}$.

By the induction hypothesis, M$[i-1,j,\ell] \cup \mathcal{A}_{r,j} \subseteq Sol_{i-1,j,\ell} \cup Sol^*_{i,j,\ell} \subseteq Sol_{i,j,\ell}$. Now, consider a pair of sets $X \in Sol_{i,j,\ell}$, and $Y \subseteq (U \setminus X)$ such that $|Y| \leq k - j$. Since $X \in Sol_{i,j,\ell}$, either $X \in Sol_{i-1,j,\ell}$ or $X \in Sol^*_{i,j,\ell}$. In the first case, the induction hypothesis implies that there is $\widehat{X} \in$ M$[i-1,j,\ell]$ such that $\widehat{X} \cap Y = \emptyset$. In the second case, since $\mathcal{A}_{r,j}$ represents $Sol^*_{i,j,\ell}$, there is $\widehat{X} \in \mathcal{A}_{r,j}$ such that $\widehat{X} \cap Y = \emptyset$. We get that M$[i-1,j,\ell] \cup \mathcal{A}_{r,j}$ represents $Sol_{i,j,\ell}$. Thus, by Observation 1, M$[i,j,\ell]$ represents $Sol_{i,j,\ell}$. □

F   Proof of Lemma 6

We use in the proof below the next result, implicitly given in [10].

Lemma 12 *For any out-tree T of G rooted at v, containing at least three nodes, in which $v = u$ or u is a leaf, there exists $C \in \mathcal{C}_{v,u}$ with whom T complies.*

Proof of Lemma 6. By Steps 1 and 2, the lemma holds for all $v, u \in V$, and $i = \ell = 0$. Now, consider some $v, u \in V$, $0 \leq i < k$, and $0 \leq \ell \leq t$ such that $1 \leq i + \ell$, and assume that the lemma holds for all $v', u' \in V$, $0 \leq i' \leq i$, and $0 \leq \ell' \leq \ell$ such that $i' + \ell' < i + \ell$.

For all $C \in \mathcal{C}_{v,u}$, let $\mathcal{T}_{v,u,i,\ell,C}$ be the set of out-trees in $\mathcal{T}_{v,u,i,\ell}$ that comply with C. Also, let $Sol_{v,u,i,\ell,C} = \{V_T \setminus \{v, u\} : T \in \mathcal{T}_{v,u,i,\ell,C}\}$. In order to prove the inductive claim, we need the following claim.

Claim 1 *For all $C \in \mathcal{C}_{v,u}$, $N[C]$ represents $Sol_{v,u,i,\ell,C}$.*

We first show that Claim 1 implies the correctness of the inductive claim. By Observation 1, it is enough to prove that $\mathcal{B} = \bigcup_{C \in \mathcal{C}_{v,u}} N[C]$ represents $Sol_{v,u,i,\ell}$. By Claim 1, $\mathcal{B} \subseteq \bigcup_{C \in \mathcal{C}_{v,u}} Sol_{v,u,i,\ell,C} \subseteq Sol_{v,u,i,\ell}$. Consider some $X \in Sol_{v,u,i,\ell}$, and $Y \subseteq (V \setminus X)$ such that $|Y| \leq k + t - (i + \ell)$. Since $X \in Sol_{v,u,i,\ell}$, there is an out-tree $T \in \mathcal{T}_{v,u,i,\ell}$ whose node set is $X \cup \{v, u\}$. By Lemma 12, there is $C \in \mathcal{C}_{v,u}$ such that $T \in \mathcal{T}_{v,u,i,\ell,C}$. We get that $X \in Sol_{v,u,i,\ell,C}$. By Claim 1, there is $\widehat{X} \in N[C] \subseteq \mathcal{B}$ such that $\widehat{X} \cap Y = \emptyset$. Thus, $\mathcal{B}$ represents $Sol_{v,u,i,\ell}$.

We now turn to the proof of Claim 1. Consider an iteration of Step 5 that corresponds to some rooted tree $C \in \mathcal{C}_{v,u}$. We first note that, formally, a subforest F' of G complies with a subforest F of C if: (1) they have the same roots, (2) $\forall v', u' \in V_F$, v' is an ancestor of u' in F iff v' is an ancestor of u' in F', (3) the leaves in C from V_F are leaves in F', and (4) in the forest obtained by removing V_F from F', each tree has at most $\frac{k+t}{d}$ nodes and at most two neighbors in F' from V_F. Recall that the set $\mathcal{F}_{v,u,C,j,i',\ell'}$ includes each subforest F' of G that complies with the subforest F of C induced by $\{w_1, \ldots, w_j\}$, such that: (1)

$V_{F'} \cap (V_C \setminus V_F) = \emptyset$, and (2) the number of internal nodes (leaves) in $V_{F'}$, excluding nodes in V_F, is i' (ℓ'). We denote $Sol_{v,u,C,j,i',\ell'} = \{V_F \setminus V_C : F \in \mathcal{F}_{v,u,C,j,i',\ell'}\}$. In order to prove Claim 1, we need the following claim.

Claim 2 *For all $1 \leq j \leq |V_C|$, $0 \leq i' \leq i^*$ and $0 \leq \ell' \leq \ell^*$, we have that $\mathrm{L}[j,i',\ell'] \subseteq Sol_{v,u,C,j,i',\ell'}$ and $\{U \cup (V_C \setminus \{v,u\}) : U \in \mathrm{L}[j,i',\ell']\}$ represents $\{U \cup (V_C \setminus \{v,u\}) : U \in Sol_{v,u,C,j,i',\ell'}\}$.*

We first show that Claim 2 implies the correctness of Claim 1. By Claim 2, $\mathrm{N}[C] = \{U \cup (V_C \setminus \{v,u\}) : U \in \mathrm{L}[|V_C|,i^*,\ell^*]\} \subseteq \{U \cup (V_C \setminus \{v,u\}) : U \in Sol_{v,u,C,|V_C|,i^*,\ell^*}\} = Sol_{v,u,i,\ell,C}$. Consider some $X \in Sol_{v,u,i,\ell,C}$, and $Y \subseteq (V \setminus X)$ such that $|Y| \leq k+t-(i+\ell)$. Since $X \in Sol_{v,u,i,\ell,C}$, $U = X \setminus V_C \in Sol_{v,u,C,|V_C|,i^*,\ell^*}$. By Claim 2, there is $\widehat{U} \in \mathrm{L}[|V_C|,i^*,\ell^*]$ such that $\widehat{U} \cap Y = \emptyset$. We get that $\widehat{X} = \widehat{U} \cup (V_C \setminus \{v,u\}) \in \mathrm{N}[C]$ and $\widehat{X} \cap Y = \emptyset$. Thus, Claim 1 is correct.

Finally, we turn to the proof of Claim 2, which by the above arguments, concludes the correctness of the lemma. By Steps 7 and 8 and the induction hypothesis for M, the claim holds for all $0 \leq i' \leq i^*$ and $0 \leq \ell' \leq \ell^*$ when $j = 1$. Now, consider some $0 \leq i' \leq i^*$, $0 \leq \ell' \leq \ell^*$ and $2 \leq j \leq |V_C|$, and assume that the claim holds for all $0 \leq i'' \leq i'$, $0 \leq \ell'' \leq \ell'$ and $1 \leq j' \leq j$ s.t. ($j' < j$ or $i''+\ell'' < i'+\ell'$). By observation 1, it is enough to prove that $\{U \cup (V_C \setminus \{v,u\}) : U \in \mathcal{A}\}$ represents $\{U \cup (V_C \setminus \{v,u\}) : U \in Sol_{v,u,C,j,i',\ell'}\}$.

Note that a set X belongs to $Sol_{v,u,C,j,i',\ell'}$ iff there are sets $U,W \subseteq X$, $0 \leq i'' \leq i'$ and $0 \leq \ell'' \leq \ell'$ satisfying $i''+\ell'' \leq \dfrac{k+t}{d}$, such that $X = U \cup W$, $U \cap (W \cup V_C) = \emptyset$ and at least one of the following conditions holds.

1. $U \in Sol_{f(w_j),w_j,i'',\ell''}$ and $W \in Sol_{v,u,C,j-1,i'-i'',\ell'-\ell''}$.
2. w_j is not a leaf in C, $\ell'' \geq 1$, $U \in Sol_{w_j,w_j,i'',\ell''}$ and $W \in Sol_{v,u,C,j,i'-i'',\ell'-\ell''}$.

Thus, by Step 10 and the inductive hypotheses for M and L, we get that $\mathcal{A} \subseteq Sol_{v,u,C,j,i',\ell'}$. Consider some $X \in Sol_{v,u,C,j,i',\ell'}$, and $Y \subseteq V \setminus (X \cup (V_C \setminus \{v,u\}))$ such that $|Y| \leq k+t-(i'+\ell' + |V_C \setminus \{v,u\}|)$. Since $X \in Sol_{v,u,C,j,i',\ell'}$, there are U, W, i'' and j'' as mentioned above. By the inductive hypotheses for M and L, there are sets $\widehat{U}$ and $\widehat{W}$ such that $\widehat{U} \cap (\widehat{W} \cup V_C) = \emptyset$ and $(\widehat{U} \cup \widehat{W}) \cap Y = \emptyset$, for which at least one of the following conditions holds.

1. $\widehat{U} \in \mathrm{M}[f(w_j),w_j,i'',\ell'']$ and $\widehat{W} \in \mathrm{L}[j-1,i' - i'',\ell' - \ell'']$.
2. w_j is not a leaf in C, $\ell'' \geq 1$, $\widehat{U} \in \mathrm{M}[w_j,w_j,i'',\ell'']$ and $\widehat{W} \in \mathrm{L}[j,i'-i'',\ell'-\ell'']$.

We get that $\widehat{X} = \widehat{U} \cup \widehat{W} \in \mathcal{A}$ and $\widehat{X} \cap Y = \emptyset$; thus, Claim 2 is correct. $\square$

G   Improving Known Applications

We now show that by simply replacing the computation of representative families of Fomin et al. [10] with our faster computation, we get improved algorithms for Long Directed Cycle, Weighted k-Path and Weighted k-Tree.

G.1   Long Directed Cycle

Given a graph $G=(V,E)$ and a parameter $k\in\mathbb{N}$, we need to decide if G contains a simple cycle of length at least k.

By using the algorithm for Long Directed Cycle given in [10] with our computation scheme, RepAlg, we immediately get that Long Directed Cycle can be solved in time

$$O(2^{o(k)}|E|\log^2|V|\cdot\max_{0\leq t\leq k}\left\{\frac{(2ck)^{2k}}{t^t(2ck-t)^{2k-t}}(\frac{2ck}{2ck-t})^{2k-t}\right\}).$$

We choose $c=1.5$. Then, the maximum is obtained at $t=k$. Thus, we solve Long Directed Cycle in time $O(6.75^k|E|\log^2|V|)$ (improving the previous $O(8^k|E|\log^2|V|)$ time).

G.2   Weighted k-Path

Given a graph $G=(V,E)$, a function $w\colon E\to\mathbb{R}$ and a parameter $k\in\mathbb{N}$, we need to find the minimal weight of a simple path of length k in G.

By using the algorithm for Weighted k-Path given in [10] with our computation scheme, RepAlg, we immediately get that Weighted k-Path can be solved in time

$$O(2^{o(k)}|V|\log^2|V|\cdot\max_{0\leq t\leq k}\left\{\frac{(ck)^k}{t^t(ck-t)^{k-t}}(\frac{ck}{ck-t})^{k-t}\right\}).$$

We choose $c=1.447$. Then, the maximum is obtained at $t=\alpha k$, where $\alpha\cong0.55277$. Thus, we solve Weighted k-Path in time $O(2.61804^k|V|\log^2|V|)$ (improving the previous $O(2.85043^k|V|\log^2|V|)$ time).

G.3   Weighted k-Tree

Given a graph $G=(V,E)$, a function $w\colon E\to\mathbb{R}$ and a tree T on k nodes, we need to find the minimal weight of a subtree of G isomorphic to T.

By using the algorithm for Weighted k-Tree given in [10] with our computation scheme, RepAlg, we immediately get that for any $\epsilon>0$, Weighted k-Tree can be solved in time

$$O(2^{\epsilon k}(\frac{1+\epsilon}{1-\epsilon})^k|V|^{O(\frac{1}{\epsilon})}\cdot\max_{0\leq t\leq k}\left\{\frac{(ck)^k}{t^t(ck-t)^{k-t}}(\frac{ck}{ck-t})^{k-t}\right\}).$$

We choose $c=1.447$. Then, the maximum is obtained at $t=\alpha k$, where $\alpha\cong0.55277$. By choosing a small enough $\epsilon>0$, we can solve Weighted k-Tree in time $O(2.61804^k|V|^{O(1)})$ (improving the previous $O(2.85043^k|V|^{O(1)})$ time).