Approximate Counting of k-Paths: Deterministic and in Polynomial Space

Andreas Björklund
Lund University, Lund, Sweden
andreas.bjorklund@cs.lth.se

Daniel Lokshtanov
University of California, Bergen, Santa Barbara, USA
daniello@ucsb.edu

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University, Beersheba, Israel
meiravze@bgu.ac.il

Abstract

A few years ago, Alon et al. [ISMB 2008] gave a *simple randomized* $\mathcal{O}((2e)^k m\epsilon^{-2})$-time exponential-space algorithm to approximately compute the number of paths on k vertices in a graph G up to a multiplicative error of $1 \pm \epsilon$. Shortly afterwards, Alon and Gutner [IWPEC 2009, TALG 2010] gave a *deterministic* exponential-space algorithm with running time $(2e)^{k+\mathcal{O}(\log^3 k)} m \log n$ whenever $\epsilon^{-1} = k^{\mathcal{O}(1)}$. Recently, Brand et al. [STOC 2018] provided a speed-up at the cost of reintroducing randomization. Specifically, they gave a *randomized* $\mathcal{O}(4^k m\epsilon^{-2})$-time exponential-space algorithm. In this article, we revisit the algorithm by Alon and Gutner. We modify the foundation of their work, and with a novel twist, obtain the following results.

- We present a *deterministic* $4^{k+\mathcal{O}(\sqrt{k}(\log^2 k+\log^2 \epsilon^{-1}))} m \log n$-time *polynomial-space* algorithm. This *matches* the running time of the best known deterministic polynomial-space algorithm for *deciding* whether a given graph G has a path on k vertices.
- Additionally, we present a *randomized* $4^{k+\mathcal{O}(\log k(\log k+\log \epsilon^{-1}))} m \log n$-time *polynomial-space* algorithm. While Brand et al. make non-trivial use of exterior algebra, our algorithm is very simple; we only make elementary use of the probabilistic method.

Thus, the algorithm by Brand et al. runs in time $4^{k+o(k)} m$ whenever $\epsilon^{-1} = 2^{o(k)}$, while our deterministic and randomized algorithms run in time $4^{k+o(k)} m \log n$ whenever $\epsilon^{-1} = 2^{o(k^{\frac{1}{4}})}$ and $\epsilon^{-1} = 2^{o(\frac{k}{\log k})}$, respectively. Prior to our work, no $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$-time *polynomial-space* algorithm was known. Additionally, our approach is embeddable in the classic framework of divide-and-color, hence it immediately extends to approximate counting of graphs of bounded *treewidth*; in comparison, Brand et al. note that their approach is limited to graphs of bounded *pathwidth*.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases parameterized complexity, approximate counting, k-Path

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.24

Category Track A: Algorithms, Complexity and Games

Funding *Saket Saurabh*: This work is supported by the European Research Council (ERC) via grant LOPPRE, reference 819416.
Meirav Zehavi: This work is supported by the Israel Science Foundation individual research grant no. 1176/18.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 24; pp. 24:1–24:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The objective of the #k-Path problem is to compute the number of k-paths – that is, (simple) paths on k vertices – in a given graph G. Unfortunately, this problem is #W[1]-hard [19], which means that it is unlikely to be solvable in time $f(k)n^{\mathcal{O}(1)}$ for any computable function f of k. Nevertheless, this problem is long known to admit an FPT-approximation scheme (FPT-AS), that is, an $f(k,\epsilon^{-1})n^{\mathcal{O}(1)}$-time algorithm that approximately computes the number of k-paths in a given graph G up to a multiplicative error of $1 \pm \epsilon$. More than 15 years ago, Arvind and Raman [6] utilized the classic method of *color coding* [5] to design a *randomized* exponential-space FPT-AS for #k-Path with running time $k^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ whenever $\epsilon^{-1} \leq k^{\mathcal{O}(k)}$. A few years afterwards, the development and use of applications in computational biology to detect and analyze *network motifs* have already become common practice [34, 37, 36, 18, 24]. Roughly speaking, a network motif is a small pattern whose number of occurrences in a given network is substantially larger than its number of occurrences in a random network. Due to their tight relation to network motifs, #k-Path and other cases of the #Subgraph Isomorphism problem became highly relevant to the study of gene transcription networks, protein-protein interaction (PPI) networks, neural networks and social networks [31]. In light of these developments, Alon et al. [2] revisited the method of color coding to attain a running time whose dependency on k is *single-exponential* rather than *slightly super-exponential*. Specifically, they designed a *simple randomized* $\mathcal{O}((2e)^k m\epsilon^{-2})$-time exponential-space FPT-AS for #k-Path, which they employed to analyze PPI networks of unicellular organisms. In particular, their algorithm has running time $2^{\mathcal{O}(k)}m$ whenever $\epsilon^{-1} \leq 2^{\mathcal{O}(k)}$.

The first *deterministic* FPT-AS for #k-Path was found in 2007 by Alon and Gutner [4]; this algorithm has an exponential space complexity and running time $2^{\mathcal{O}(k \log\log k)}m \log n$ whenever $\epsilon^{-1} = 2^{o(\log k)}$. Shortly afterwards, Alon and Gutner [3] improved upon their previous work, and designed a deterministic exponential-space FPT-AS for #k-Path with running time $(2e)^{k+\mathcal{O}(\log^3 k)}m \log n$ whenever $\epsilon^{-1} = k^{\mathcal{O}(1)}$. For close to a decade, this algorithm has remained the state-of-the-art. In contrast, during this decade, the k-Path problem (the decision version of #k-Path) has seen several improvements that were considered to be breakthroughs at their time [14, 26, 8, 10, 21]. In 2016, Koutis and Williams [27] conjectured that #k-Path admits an FPT-AS with running time $2^k n^{\mathcal{O}(1)}$. Recently, at the cost of reintroducing randomization, Brand et al. [13] provided a speed-up towards the resolution of this conjecture. Specifically, they gave an algebraic *randomized* $\mathcal{O}(4^k m\epsilon^{-2})$-time exponential-space algorithm. In the context of Parameterized Complexity in general, and the k-Path problem in particular, the power of randomization is an issue of wide interest [1]. Specifically for the k-Path problem, an algebraic randomized $2^k n^{\mathcal{O}(1)}$-time algorithm has been found already a decade ago [38], and since then, the existence of a deterministic algorithm that exhibits the same time complexity has been repeatedly posed as a major open problem in the field. Both Koutis and Williams conjectured this question to have an affirmative answer in several venues [38, 28, 27]. Clearly, this question is simpler than the one of the design of a deterministic FPT-AS for #k-Path with running time $2^k n^{\mathcal{O}(1)}$.

In this article, we modify the foundation of the work of Alon and Gutner [4, 3], and with a novel twist, obtain the following results (see Theorem 21 and Corollary 10).

- First, we present a *randomized* $4^{k+\mathcal{O}(\log k(\log k + \log \epsilon^{-1}))}m \log n$-time *polynomial-space* algorithm. While Brand et al. [13] make non-trivial use of exterior algebra, *our randomized algorithm is very simple*: we only make elementary use of the probabilistic method.[1]

[1] Of course, simplicity is a subjective matter, which may depend on the background of the reader.

- Additionally, we present a *deterministic* $4^{k+\mathcal{O}(\sqrt{k}(\log^2 k+\log^2 \epsilon^{-1}))} m \log n$-time *polynomial-space* algorithm. In particular, *without compromising time complexity, we attain both the properties of having a polynomial space complexity and being deterministic simultaneously.* In fact, even though we deal with #k-Path, the running time of our algorithm *matches* the best known running time of a deterministic polynomial-space algorithm for k-Path (the decision version of #k-Path) [14].

Thus, the algorithm by Brand et al. [13] runs in time $4^{k+o(k)}m$ whenever $\epsilon^{-1} = 2^{o(k)}$, while our deterministic and randomized algorithms run in time $4^{k+o(k)}m \log n$ whenever $\epsilon^{-1} = 2^{o(k^{\frac{1}{4}})}$ and $\epsilon^{-1} = 2^{o(\frac{k}{\log k})}$, respectively.

Prior to our work, no $c^k n^{\mathcal{O}(1)}$-time *polynomial-space* (even randomized) algorithm for #k-Path was known *for any constant c*. The design of polynomial-space parameterized algorithms is an active research area in Parameterized Complexity. Even (sometimes) at a notable compromise of time complexity, the property of having polynomial space complexity is sought (see, e.g., [20, 30, 29, 7, 23]). Indeed, algorithms with high space complexity are in practice more constrained because the amount of memory is not easily scaled beyond hardware constraints whereas time complexity can be alleviated by allowing for more time for the algorithm to finish. Furthermore, algorithms with low space complexity are typically easier to parallelize and more cache-friendly.

Additionally, our approach is embeddable in the classic framework of divide-and-color, hence it immediately extends to approximate counting of graphs of bounded treewidth;in comparison, Brand et al. [13] note that their approach is limited to graphs of bounded pathwidth. Similarly, we can approximately count various other objects such as q-dimensional p-matchings, q-set p-packings, graph motifs, and more:

▶ **Theorem 1.** *The following problems admit deterministic $4^{k+\mathcal{O}(\sqrt{k}(\log^2 k+\log^2 \frac{1}{\epsilon}))} n^{\mathcal{O}(1)}$-time (resp. randomized $4^{k+\mathcal{O}(\log^2 k)}(\frac{1}{\epsilon})^{\mathcal{O}(\log k)} n^{\mathcal{O}(1)}$-time) FPT-ASs with polynomial space complexity:* (i) #Subgraph Isomorphism *for k-vertex subgraphs of treewidth $\mathcal{O}(1)$;* (ii) #q-Dimensional p-Matching *with $k = (q-1)p$;* (iii) #q-Set p-Packing *with $k = qp$;* (iv) #Graph Motif *and* #Module Motif *with $k = 2p$ where p is the motif size;* (v) #p-Internal Out-Branching *with $k = 2p$;* (vi) #Partial Cover *for k-element solutions.*[2]

Towards the design of our algorithms, our first conceptual contribution is the introduction of the notion of an *approximate parsimonious splitter*. While a randomized construction of such an object is simple, we do not know how (or whether it is even possible) to compute it deterministically within the size and time bounds that we require. We believe that this gap in knowledge of derandomization is the main reason why, for close to a decade, no progress has been made upon the result by Alon and Gutner [4, 3]. Here, our second conceptual contribution comes into play. We show that for recursive procedures, a weaker object that can only split so called nice sets suffices, since the recursion itself can keep track on the "niceness" of sets. We believe that both the concept of approximate parsimonious splitters as well as our approach of how to weaken a randomized object (to efficiently compute it deterministically) at the cost of simple bookkeeping might find further applications in the future. Our ideas and methods are discussed in more detail in Section 3.

[2] For problems (i) and (iv), the basis 4 is replaced by the basis 4.001 (or, more precisely, $4+\delta$ for any fixed constant $\delta > 0$).

Table 1 State-of-the-art of $\#k$-Path and k-Path.

Ref.	Time	Counting	Deterministic	Poly. Space	Extension
[14]	$4^{k+o(k)}n^{\mathcal{O}(1)}$	No	Yes	Yes	Treewidth $\mathcal{O}(1)$
[40]	$2.597^k n^{\mathcal{O}(1)}$	No	Yes	No	Treewidth $\mathcal{O}(1)$
[38]	$2^k n^{\mathcal{O}(1)}$	No	No	Yes	Treewidth $\mathcal{O}(1)$
[10]	$1.657^k n^{\mathcal{O}(1)}$	No	No	Yes	No Extension
[3]	$(2e)^{k+o(k)}n^{\mathcal{O}(1)}$	Yes	Yes	No	Treewidth $\mathcal{O}(1)$
[13]	$4^k n^{\mathcal{O}(1)}$	Yes	No	No	Pathwidth $\mathcal{O}(1)$
This Paper	$4^{k+o(k)}n^{\mathcal{O}(1)}$	Yes	Yes	Yes	Treewidth $\mathcal{O}(1)$

Related Work. The algorithms by Alon et al. [2] and Alon and Gutner [4, 3], just like our algorithms, extend to approximate counting of graphs of bounded treewidth. (This remark is also made by Alon and Gutner [4, 3].) In what follows, we briefly review works related to exact counting and decision from the viewpoint of Parameterized Complexity. Since these topics are not the focus of our work, the survey is illustrative rather than comprehensive.

The problem of counting the number of subgraphs of a graph G that are isomorphic to a graph H – that is, #Subgraph Isomorphism with Pattern H – admits a dichotomy: If the vertex cover number of H is bounded, then it is FPT [39], and otherwise it is #W[1]-hard [16]. The #W[1]-hardness of $\#k$-Path, originally shown by Flum and Grohe [19], follows from this dichotomy. By using the "meet in the middle" approach, the $\#k$-Path problem and, more generally, #Subgraph Isomorphism with Pattern H where H has bounded *pathwidth* and k vertices, was shown to admit an $n^{\frac{k}{2}+\mathcal{O}(1)}$-time algorithm [9]. Later, Björklund et al. [12] showed that $\frac{k}{2}$ is not a barrier (which was considered to be the case at that time) by designing an $n^{0.455k+\mathcal{O}(1)}$-time algorithm. Recently, a breakthrough that resulted in substantially faster running times took place: Curticapean et al. [15] showed that #Subgraph Isomorphism with Pattern H is solvable in time $\ell^{\mathcal{O}(\ell)}n^{0.174\ell}$ where ℓ is the number of edges in H; in particular, this algorithm solves $\#k$-Path in time $k^{\mathcal{O}(k)}n^{0.174k}$.

The k-Path problem (on both directed and undirected graphs) is among the most extensively studied parameterized problems [17, 22]. After a long sequence of works in the past three decades, the current best known parameterized algorithms for k-Path have running times $1.657^k n^{\mathcal{O}(1)}$ (randomized, polynomial space, undirected only) [10, 8] (extended in [11]), $2^k n^{\mathcal{O}(1)}$ (randomized, polynomial space) [38], $2.597^k n^{\mathcal{O}(1)}$ (deterministic, exponential space) [40, 21, 35], and $4^{k+o(k)}n^{\mathcal{O}(1)}$ (deterministic, polynomial space) [14]. The $1.657^k n^{\mathcal{O}(1)}$-time algorithm of Björklund et al. [10, 8] crucially relies on the symmetric structure of undirected k-paths. However, all other algorithms above directly extend to the detection of subgraphs of bounded treewidth. In particular, if the running time of the algorithm is $c^k n^{\mathcal{O}(1)}$, then the running time of the extension is $c^k n^{t+\mathcal{O}(1)}$ where t is the treewidth of the sought graph. To ensure that the constant c remains the same when dealing with the two deterministic algorithms (of [40, 21, 35] and [14]), the "division into small trees" trick by Fomin et al. [21] can be used; for the randomized algorithm (of [38]), no trick is required.

2 Preliminaries

For the sake of readability, we ignore ceiling and floor signs. Given a graph G, we let $V(G)$ and $E(G)$ denote the vertex set and edge set of G, respectively. For a positive integer k, a *k-path* in G is a (simple) path on k vertices in G; in case G is directed, the path is directed as well. We let $n = |V(G)|$ and $m = |E(G)|$. For a subset $U \subseteq V(G)$, $G[U]$ denotes the subgraph of U induced by G, and $G - U = G[V(G) \setminus U]$.

For a function $f : A \to B$ and subsets $A' \subseteq A$ and $B' \subseteq B$, define $f(A') = \{f(a) : a \in A'\}$ and $f^{-1}(B') = \{a \in A : f(a) \in B'\}$. For two functions $f : A \to B$ and $g : B \to C$, the notation $g \circ f : A \to C$ refers to function composition. For two tuples $X = (x_1, x_2, \dots, x_p)$ and $Y = (y_1, y_2, \dots, y_q)$, denote their concatenation by $X \diamond Y = (x_1, x_2, \dots, x_p, y_1, y_2, \dots, y_q)$. By standard Chernoff bounds, we have the following bounds.

▶ **Proposition 2** ([32]). *Let $X_1, \dots, X_n$ be independent random variables, each assigned a value in $\{0, 1\}$. Let $X = \sum_{i=1}^{n} X_i$, and let $\mu = E[X]$ denote the expected value of X. Then, for any $0 \leq \delta \leq 1$, it holds that* (i) $\Pr(X \leq (1-\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2}}$ *and* (ii) $\Pr(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2\mu}{3}}$.

Universal Families. For any $k \in \mathbb{N}$, a *k-set* is a set of size k. Given a universe U, denote $\binom{U}{k} = \{S \subseteq U : |S| = k\}$. Given a family $\mathcal{F}$ over U and two subsets $A, B \subseteq U$, denote $\mathcal{F}[A, B] = \{F \in \mathcal{F} : A \subseteq F, B \cap F = \emptyset\}$. Next, we present the definition of a universal family.

▶ **Definition 3** (Universal Family [33, 21]). *Let $n, p, q \in \mathbb{N}$. A family $\mathcal{F}$ of sets over a universe U of size n is an (n, p, q)-*universal family *if for each pair of disjoint sets $A \in \binom{U}{p}$ and $B \in \binom{U}{q}$, there is a set $F \in \mathcal{F}$ that contains A and is disjoint from B, that is, $\mathcal{F}[A, B] \neq \emptyset$.*

In the classic setting by Naor et al. [33], $p = q$. However, as shown by Fomin et al. [21], cases where $p \neq q$ are also of interest. Specifically, the following well-known proposition asserts that small representative families can be computed efficiently.

▶ **Proposition 4** ([33, 21]). *Let $n, p, q \in \mathbb{N}$, and $k = p + q$. Let U be a universe of size n. Then, an (n, p, q)-universal family $\mathcal{F}$ of sets over U of size $\mathcal{O}(\binom{k}{p} \log n)$ can be computed with success probability $1 - 1/n$ in time $\mathcal{O}(\binom{k}{p} n \log n)$. Additionally, an (n, p, q)-universal family $\mathcal{F}$ of sets over U of size $\binom{k}{p} 2^{o(k)} \log n$ can be computed (deterministically) in time $\binom{k}{p} 2^{o(k)} n \log n$. Both computations can enumerate the sets in $\mathcal{F}$ with polynomial delay.*

Observe that the constructions above are essentially optimal since any (n, p, q)-universal family must be of size at least $\binom{k}{p}$. We later extend Definition 3 to be approximately parsimonious, and show how to compute approximate parsimonious universal families.

3 Overview of Our Ideas and Methods

In this section, we discuss our main ideas and methods. Additionally, we present a simplified version of one of our applications in detail.

3.1 Approx. Parsimonious Universal Family: Randomized Construction

For any pair of disjoint sets $A \in \binom{U}{p}$ and $B \in \binom{U}{q}$, Definition 3 guarantees that $\mathcal{F}[A, B] \neq \emptyset$. However, the number of sets in $\mathcal{F}[A, B]$ can be arbitrary. In our applications, the number of sets in $\mathcal{F}[A, B]$ will be tightly linked to the number of solutions whose "first half" is in A and whose "second half" is in B; thus, to avoid over-counting some solutions, we need all families $\mathcal{F}[\cdot, \cdot]$ to be *roughly* of the same size. For this purpose, let us first extend Definition 3 to be approximately parsimonious.

▶ **Definition 5** (δ-Parsimonious Universal Family). *Let $n, p, q \in \mathbb{N}$ and $0 < \delta < 1$. Denote $k = p + q$. A family $\mathcal{F}$ of sets over a universe U of size n is a δ-*parsimonious (n, p, q)-universal family *if there exists $T = T(n, p, q, \delta) > 0$ such that for each pair of disjoint sets $A \in \binom{U}{p}$ and $B \in \binom{U}{q}$, it holds that $(1 - \delta) \cdot T \leq |\mathcal{F}[A, B]| \leq (1 + \delta) \cdot T$.*

We call the value T above a *correction factor*, and suppose it to be given along with the family $\mathcal{F}$. Our randomized computation of a δ-parsimonious (n, p, q)-universal family is based on the probabilistic method, inspired by [33, 21]. Specifically, we prove the following.

▶ **Theorem 6.** *Let $n, p, q \in \mathbb{N}$ and $0 < \delta < 1$, and denote $k = p + q$. Let U be a universe of size n. A δ-parsimonious (n, p, q)-universal family $\mathcal{F}$ of sets over U of size $t = \mathcal{O}\left(\frac{k^k}{p^p q^q} \cdot k \log n \cdot \frac{1}{\delta^2}\right)$,[3] can be computed with success probability at least $1 - 1/n^{100k}$ in time $\mathcal{O}(t \cdot n)$. In particular, the sets in $\mathcal{F}$ can be enumerated with delay $\mathcal{O}(n)$.*

We note that the choice of 100 is arbitrary; it can be replaced by the choice of any fixed constant c. Crucially, we gain the extra property of being δ-parsimonious while essentially having the same time complexity and upper bound on the size of the output as in the non-parsimonious construction.

3.2 Warm Up Application: Simple Randomized FPT-AS for $\#k$-Path

Before we delve into more technical and less intuitive definitions related to our deterministic construction, we find it important to understand the relation between Definition 5 and $\#k$-Path. For this purpose, we present a simple randomized polynomial-space FPT-AS for $\#k$-Path. The dependency of the time complexity on n is made almost linear in Section 3.3). While the improved algorithm is still short and simple, it is somewhat less intuitive and hence presented separately later. For the sake of illustration, suppose that G is undirected.

Algorithm. Let $\widehat{\epsilon} = \ln(1 + \epsilon)$ and $\epsilon' = \widehat{\epsilon}/(k-1)$. Our algorithm is a recursive algorithm, denoted by $\mathcal{A}$. Each call to $\mathcal{A}$ is of the form $\mathcal{A}(G', k')$ where G' is an induced subgraph of G and $k' \in \{1, \ldots, k\}$. For all $u, v \in V(G')$, the call $\mathcal{A}(G', k')$ should output an integer $a_{u,v}$ that approximates the number of k'-paths with endpoints u and v in G'. The initial call to the algorithm is with $G' = G$ and $k' = k$, and the final output is $(\sum_{u,v \in V(G)} a_{u,v})/2$.

We turn to describe a call $\mathcal{A}(G', k')$. In the basis, where $k' = 1$, we return $a_{v,v} = 1$ for all $v \in V(G')$, and $a_{u,v} = 0$ for all $u, v \in V(G')$ (with $u \neq v$).

Now, suppose that $k' \geq 2$. By Theorem 6, for an ϵ'-parsimonious $(n, k'/2, k'/2)$-universal family $\mathcal{F}$ of sets over $V(G)$, we can enumerate the sets $F \in \mathcal{F}$ with delay $\mathcal{O}(n)$. For each set $F \in \mathcal{F}$, we proceed as follows. We first perform two recursive calls: *(i)* we call $\mathcal{A}$ with $(G'[F], k'/2)$; *(ii)* we call $\mathcal{A}$ with $(G' - F, k'/2)$. For any $u, v \in F \cap V(G')$, let $b^F_{u,v}$ denote the number returned by the first call. Similarly, for any $u, v \in V(G') \setminus F$, let $c^F_{u,v}$ denote the number returned by the second call. Then, for all $u \in F$ and $v \in V(G') \setminus F$, define $a^F_{u,v} = \sum_{\substack{\{p,q\} \in E(G') \\ \text{s.t. } p \in F, q \notin F}} b^F_{u,p} \cdot c^F_{q,v}$.

Let T be the correction factor of $\mathcal{F}$. After all sets $F \in \mathcal{F}$ were enumerated, for all $u, v \in V(G')$, we output $a_{u,v}$ calculated as follows: $a_{u,v} = \frac{1}{T} \cdot \sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } u \in F, v \notin F}} a^F_{u,v}$. Note that we do not store all the values $a^F_{u,v}$ simultaneously, but we merely store one such value at a time and delete it immediately after $a^F_{u,v}/T$ is added. This completes the description of $\mathcal{A}$.

[3] Note that as $p + q = k$, the value $\frac{k^k}{p^p q^q}$ is upper bounded by 2^k rather than being of the magnitude of k^k.

Analysis. The main part of the analysis is done in the proof of the following lemma.

▶ **Lemma 7.** *For some fixed constant $\eta > 0$, any call $\mathcal{A}(G',k')$ has polynomial space complexity and running time $\eta^{\log k'} 4^{k'} {k'}^{\log k'} (\log n)^{\log k'} mn^2 (\frac{1}{\epsilon'^2})^{\log k'}$. Additionally, if all constructions of approximate universal families were successful, then for all $u,v \in V(G')$, the number $a_{u,v}$ returned by $\mathcal{A}(G',k')$ satisfies $(1-\epsilon')^{k'-1} x_{u,v} \leq a_{u,v} \leq (1+\epsilon')^{k'-1} x_{u,v}$ where $x_{u,v}$ is the number of k'-paths with endpoints u and v in G'.*

Proof. Let $k' = k/2^d$. We choose $\eta = 10\max\{\lambda,\tau\}$, where λ and τ are fixed constants defined later. The proof is by backwards induction of d. In the basis ($k'=1$), the claim is trivial. Now, let $d \leq \log_2 k - 1$, and suppose that the claim holds for $d+1$. Clearly, $\mathcal{A}(G',k')$ has a polynomial space complexity. By Theorem 6, for a fixed constant $\lambda > 0$ (that is independent of η),

$$|\mathcal{F}| \leq \lambda \cdot \frac{{k'}^{k'}}{(k'/2)^{k'/2}(k'/2)^{k'/2}} \cdot k' \log n \cdot \frac{1}{\epsilon'^2} = \lambda \cdot 2^{k'} \cdot k' \log n \cdot \frac{1}{\epsilon'^2}.$$

Moreover, by the inductive hypothesis, for a fixed constant $\tau > 0$, the running time of $\mathcal{A}(G',k')$ is upper bounded by $|\mathcal{F}| \cdot \left(2 \cdot \eta^{\log \frac{k'}{2}} 4^{\frac{k'}{2}} (\frac{k'}{2})^{\log \frac{k'}{2}} (\log n)^{\log \frac{k'}{2}} mn^2 (\frac{1}{\epsilon'^2})^{\log \frac{k'}{2}} + \tau mn^2\right)$. Note that τ is independent of η. By choosing $\eta = 10\max\{\lambda,\tau\}$, this means that the running time of $\mathcal{A}(G',k')$ is upper bounded by

$$\begin{aligned}
& |\mathcal{F}| \cdot \left(2 \cdot \eta^{\log \frac{k'}{2}} 4^{\frac{k'}{2}} (\frac{k'}{2})^{\log \frac{k'}{2}} (\log n)^{\log \frac{k'}{2}} mn^2 (\frac{1}{\epsilon'^2})^{\log \frac{k'}{2}} + \tau mn^2\right) \\
\leq\ & \frac{\eta}{10} 2^{k'} k' \log n \frac{1}{\epsilon'^2} \cdot \left(2 \cdot \eta^{\log k' - 1} 2^{k'} {k'}^{\log k' - 1} (\log n)^{\log k' - 1} mn^2 (\frac{1}{\epsilon'^2})^{\log k' - 1} + \frac{\eta}{10} mn^2\right) \\
\leq\ & \eta^{\log k'} 4^{k'} {k'}^{\log k'} (\log n)^{\log k'} mn^2 (\frac{1}{\epsilon'^2})^{\log k'}.
\end{aligned}$$

This completes the proof of the first item of the claim.

Towards the proof of the second item of the claim, suppose that all constructions of approximate universal families were successful, and consider some $u,v \in V(G')$. Let $x^{\widehat{G}}_{p,q}$ denote the number of $k'/2$-paths with endpoints p and q in $\widehat{G}$. By the inductive hypothesis, we have that

$$\begin{aligned}
a_{u,v} &= \frac{1}{T} \cdot \sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } u \in F, v \notin F}} a^F_{u,v} = \frac{1}{T} \cdot \sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } u \in F, v \notin F}} \left(\sum_{\substack{\{p,q\} \in E(G') \\ \text{s.t. } p \in F, q \notin F}} b^F_{u,p} \cdot c^F_{q,v} \right) \\
&\leq \frac{1}{T} \cdot \sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } u \in F, v \notin F}} \left(\sum_{\substack{\{p,q\} \in E(G') \\ \text{s.t. } p \in F, q \notin F}} (1+\epsilon')^{\frac{k'}{2}-1} x^{G'[F]}_{u,p} \cdot (1+\epsilon')^{\frac{k'}{2}-1} x^{G'-F}_{q,v} \right) \\
&= \frac{1}{T} \cdot (1+\epsilon')^{k'-2} \cdot \sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } u \in F, v \notin F}} \left(\sum_{\substack{\{p,q\} \in E(G') \\ \text{s.t. } p \in F, q \notin F}} x^{G'[F]}_{u,p} \cdot x^{G'-F}_{q,v} \right)
\end{aligned}$$

Let $\mathcal{P}_{u,v}$ denote the set of k'-paths in G' with endpoints u and v. In addition, for any subset $F \subseteq V(G')$, let $\mathcal{P}_{u,v}[F]$ denote the set of paths $P \in \mathcal{P}_{u,v}$ where the $k'/2$ vertices on P closest to u (including u) belong to F and the other $k'/2$ vertices on P do not belong to F. Thus,

$$a_{u,v} \leq (1+\epsilon')^{k'-2} \cdot \frac{\sum_{F \in \mathcal{F}} |\mathcal{P}_{u,v}[F]|}{T}.$$

Since $\mathcal{F}$ is an ϵ'-parsimonious $(n, k'/2, k'/2)$-universal family, for any path $P \in \mathcal{P}_{u,v}$ it holds that the number of sets $F \in \mathcal{F}$ such that $P \in \mathcal{P}_{u,v}[F]$ is upper bounded by $(1+\epsilon')T$. Thus,

$$a_{u,v} \leq (1+\epsilon')^{k'-2} \cdot \frac{(1+\epsilon')T|\mathcal{P}_{u,v}|}{T} = (1+\epsilon')^{k'-1} x_{u,v}.$$

Symmetrically, we derive that $(1-\epsilon')^{k'-1} x_{u,v} \leq a_{u,v}$. This completes the proof. ◀

We now conclude the following theorem.

▶ **Theorem 8.** *There is a randomized* $(4^{k+o(k)}mn^2 + mn^{2+o(1)})(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}$*-time polynomial-space algorithm that, given a graph G, a positive integer k and an accuracy value* $0 < \epsilon < 1$, *outputs a number y that (with high probability, say, at least* $9/10$*) satisfies* $(1-\epsilon)x \leq y \leq (1+\epsilon)x$ *where x is the number of k-paths in G. In particular, if* $\frac{1}{\epsilon} = 2^{o(k/\log k)}$*, then the running time is* $4^{k+o(k)}mn^2 + mn^{2+o(1)}$.

Proof. By Lemma 7 with $G' = G$ and $k' = k$, we know that the total running time of $\mathcal{A}(G,k)$ is bounded by $4^{k+\mathcal{O}(\log^2 k)}(\log n)^{\log k} mn^2 (\frac{1}{\epsilon'})^{\log k}$ and uses polynomial space. Additionally, if all constructions of approximate universal families were successful, then for all $u, v \in V(G)$, the number $a_{u,v}$ computed by $\mathcal{A}(G,k)$ satisfies $(1-\epsilon')^{k-1} x_{u,v} \leq a_{u,v} \leq (1+\epsilon')^{k-1} x_{u,v}$ where $x_{u,v}$ is the number of k-paths with endpoints u and v in G.

If $\log n \leq 2^{\sqrt{k}}$, then $(\log n)^{\log k} \leq 2^{o(k)}$. Otherwise, when $\log n > 2^{\sqrt{k}}$, it holds that $k < \log^2 \log n$. It follows that $4^{k+\mathcal{O}(\log^2 k)}(\log n)^{\log k} \leq 4^{\log^2 \log n + \mathcal{O}(\log\log\log n)}(\log n)^{2\log\log\log n} \leq n^{\mathcal{O}(\frac{\log^2 \log n}{\log n})} \leq n^{o(1)}$. In addition, by Taylor series $\ln(1+x) = \sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^n}{n}$, it follows that $\epsilon/2 \leq \epsilon - \epsilon^2/2 \leq \ln(1+\epsilon) = \widehat{\epsilon} \leq \epsilon$, which means that $(\frac{1}{\epsilon'})^{\log k} = 2^{\mathcal{O}(\log^2 k)}(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}$. Thus, $4^{k+\mathcal{O}(\log^2 k)}(\log n)^{\log k} mn^2 (\frac{1}{\epsilon'})^{\log k} = (4^{k+o(k)}mn^2 + mn^{2+o(1)})(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}$.

We now claim that with high probability, all constructions of approximate universal families were successful. By Theorem 6, the probability that a single construction is successful is at least $1 - 1/n^{100k}$. Thus, the probability that all constructions are successful is at least $(1-1/n^{100k})^{\mu}$ where μ is the number of constructions. Clearly, the number of constructions is upper bounded by the running time of $\mathcal{A}$. In turn, we can assume w.l.o.g. that the upper bound proven on this running time is, in itself, upper bounded by n^k, since otherwise the problem can be solved exactly by brute force within it. Thus, $\mu \leq n^k$. From this, we know that the probability that all constructions are successful is at least $(1-1/n^{100k})^{n^k}$. As n grows larger, this value approaches 1. In particular, the success probability can be assumed to be at least $9/10$ (otherwise n is a fixed constant), which proves our claim.

Thus, we know that for all $u, v \in V(G)$, it holds that $(1-\epsilon')^{k-1} x_{u,v} \leq a_{u,v} \leq (1+\epsilon')^{k-1} x_{u,v}$. Substituting ϵ' by $\widehat{\epsilon}$, we have that for all $u, v \in V(G)$, it holds that $(1-\widehat{\epsilon})x_{uv} \leq (1-\frac{\widehat{\epsilon}}{k-1})^{k-1} x_{u,v} \leq a_{u,v} \leq (1+\frac{\widehat{\epsilon}}{k-1})^{k-1} x_{u,v} \leq e^{\widehat{\epsilon}} x_{u,v}$. Since $(1-\epsilon) \leq (1-\widehat{\epsilon})$ and $e^{\widehat{\epsilon}} = (1+\epsilon)$, we have that for all $u, v \in V(G)$, it holds that $(1-\epsilon)x_{u,v} \leq a_{u,v} \leq (1+\epsilon)x_{u,v}$. Thus,

$$\begin{aligned} y &= \left(\sum_{u,v \in V(G)} a_{u,v} \right) /2 \leq \left(\sum_{u,v \in V(G)} (1+\epsilon) x_{u,v} \right) /2 \\ &= (1+\epsilon) \left(\sum_{u,v \in V(G)} x_{u,v} \right) /2 = (1+\epsilon)x. \end{aligned}$$

Symmetrically, we obtain that $(1-\epsilon)x \leq y$. This completes the proof. ◀

3.3 Improved Randomized FPT-AS for $\#k$-Path

As our improved randomized FPT-AS is less intuitive, we first discuss the intuition behind it. Here, in addition to G' and k', every call to the recursive algorithm $\mathcal{A}$ is given an assignment $\alpha' : V(G) \setminus V(G') \rightarrow \mathbb{N}_0$ of a non-negative integer to each vertex outside G'. Roughly speaking, for each vertex $v \in V(G) \setminus V(G')$, the value $\alpha'(v)$ is an approximation of the number of $\widehat{k}$-paths that end at v and are completely contained in $G - U$ for a certain integer $\widehat{k} \in \{1, 2, \dots, k - k'\}$ and a subset $U \subseteq V(G)$ that contains $V(G')$. In particular, given that now the goal of each call is to output such an assignment for $G - (U \setminus V(G')$ (a precise definition of the goal of a call is given in the formal description of the algorithm), we do not need to consider every pair of vertices $u, v \in V(G')$ and compute a value $a_{u,v}$; instead, we only compute one value per vertex. Additionally, recall that in the previous algorithm in order to compute $a_{u,v}$, we considered every edge $\{p, q\} \in E(G')$ while computing $a^F_{u,v}$ and hence divided our task into the computation of $k'/2$-paths between u and p in one recursive call and $k'/2$-paths between q and u in the other. Here, we do not store the two endpoints of paths, but their "middle". More precisely, the flow of information differs: to compute the assignment we need to output in the current call, we perform one recursive call to which the assignment α' is given as input; this call will return an assignment that "handles" the first $\widehat{k} + k'/2$ vertices on the paths being counted, and be sent as input to the second recursive call to handle the next $k'/2$ vertices.

Algorithm. Let $\widehat{\epsilon} = \ln(1 + \epsilon)$ and $\epsilon' = \widehat{\epsilon}/(k - 1)$. We add a new vertex s to G and connect it to all vertices in G. Thus, rather than counting the number of k-paths in the former graph G, we can count the number of $(k + 1)$-paths with s as an endpoint in the new graph G. In what follows, we focus on this goal.

Our algorithm is a recursive algorithm, denoted by $\mathcal{A}$. Each call to $\mathcal{A}$ is of the form $\mathcal{A}(G', k', \alpha')$ where G' is an induced subgraph of G, $k' \in \{1, \dots, k\}$, and $\alpha' : V(G) \setminus V(G') \rightarrow \mathbb{N}_0$. The call $\mathcal{A}(G', k', \alpha')$ should output an assignment $\alpha : V(G') \rightarrow \mathbb{N}_0$ with the following property: For each vertex $v \in V(G')$, it holds that $\alpha(v)$ approximates the following number:

$$\sum_{\substack{\{p,q\} \in E(G) \\ \text{s.t. } p \notin V(G'), q \in V(G')}} \alpha'(p) \cdot x_{q,v},$$

where $x_{q,v}$ is the number of k'-paths in G' between q and v.

The initial call to the algorithm is with $G' = G - \{s\}$, $k' = k$, and $\alpha'(s) = 1$. The final output is $\sum_{v \in V(G) \setminus \{s\}} \alpha(v)$.

We turn to describe a call $\mathcal{A}(G', k', \alpha')$. In the basis, where $k' = 1$, we return an assignment $\alpha : V(G') \rightarrow \mathbb{N}_0$ defined as follows: For each vertex $v \in V(G')$, define

$$\alpha(v) = \sum_{\substack{u \notin V(G') \\ \text{s.t. } \{u,v\} \in E(G)}} \alpha'(u).$$

Now, suppose that $k' \geq 2$. By Theorem 6, for an ϵ'-parsimonious $(n, k'/2, k'/2)$-universal family $\mathcal{F}$ of sets over $V(G)$, we can enumerate the sets $F \in \mathcal{F}$ with delay $\mathcal{O}(n)$. For each set $F \in \mathcal{F}$, we proceed as follows. We first recursively call $\mathcal{A}$ with $(G'[F], k'/2, \alpha')$ where α' is extended to assign 0 to every vertex in $V(G') \setminus F$. Let $\widehat{\alpha}_F$ be the output of this call, and extend it to assign 0 to every vertex in $V(G) \setminus V(G')$. Then, we recursively call $\mathcal{A}$ with $(G' - F, k'/2, \widehat{\alpha}_F)$. Let α_F be the output of this recursive call.

Let T be the correction factor of $\mathcal{F}$. After all sets $F \in \mathcal{F}$ were enumerated, the output $\alpha : V(G') \to \mathbb{N}_0$ is computed as follows. For all $v \in V(G')$, we calculate

$$\alpha(v) = \left(\sum_{\substack{F \in \mathcal{F} \\ \text{s.t. } v \notin F}} \alpha_F(v) \right) / T.$$

Note that we do not store all the assignments α_F simultaneously, but we merely store one such assignment at a time and delete it immediately after $\alpha_F(v)/T$, for every $v \in V(G')$, is added. This completes the description of $\mathcal{A}$.

Correctness. The proof of correctness of our algorithm roughly follows the same lines as the proof of correctness of Theorem 8. Due to space constraints, we omit the details, and conclude this section with the statement of our result.

▶ **Theorem 9.** *There is a randomized $(4^{k+o(k)}m + mn^{o(1)})(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}$-time polynomial-space algorithm that, given a graph G, a positive integer k and an accuracy value $0<\epsilon<1$, outputs a number y that (with high probability) satisfies $(1-\epsilon)x \leq y/2 \leq (1+\epsilon)x$ where x is the number of k-paths in G. In particular, if $\frac{1}{\epsilon} = 2^{o(k/\log k)}$, then the running time is $4^{k+o(k)}mn^{o(1)}$.*

Additionally, we can obtain the following corollary. (This corollary does not follow directly from Theorem 9, but requires a simple preliminary step to shrink the universe; due to space constraints, the details are omitted.)

▶ **Corollary 10.** *There is a randomized $4^{k+\mathcal{O}(\log^2 k)}m \log n(\frac{1}{\epsilon})^{\mathcal{O}(\log k)}$-time polynomial-space algorithm that, given a graph G, a positive integer k and an accuracy value $0<\epsilon<1$,‘outputs a number y that (with high probability) satisfies $(1-\epsilon)x \leq y/2 \leq (1+\epsilon)x$ where x is the number of k-paths in G. In particular, if $\frac{1}{\epsilon} = 2^{o(k/\log k)}$, then the running time is $4^{k+o(k)}m \log n$.*

3.4 Approx. Parsimonious Universal Family: Deterministic Construction

We do not know how to deterministically construct small δ-parsimonious universal families. Indeed, the best construction that we are aware of is the one based on bipartite Paley graphs (see Theorem 11.9 in the book by Jukna [25] and the historical notes behind the result). This construction leads to families of size $4^{k+o(k)}$ for $p = q = \frac{k}{2}$, whereas we would like size $2^{k+o(k)}$. Instead, we provide an efficient deterministic computation of a small δ-parsimonious universal family that is suitable for handling so called "nice pairs". The crucial point is that with respect to our applications, this relaxed construction suffices. In this section, we present the definition of this relaxation, its construction and main property. Due to space constraints, the proofs of the two lemmas and the theorem stated in this section are omitted.

To simplify the following definitions, we introduce the following notation. To see the intuition behind this notation in the context of applications, throughout this section h can be thought of as a function that reduces the size of the universe from n to z, f can be thought of as a function that splits the reduced universe into t parts, and $\overline{\mathbf{p}}$ can be thought of as a function that tells us that each part has k/t "useful" elements (e.g., vertices of paths to be counted in a certain recursive call) among which either p_i or $(k/t) - p_i$ were "exhausted".

▶ **Definition 11.** *Let $n, p, q, t, z \in \mathbb{N}$, and $k = p+q$. Let U be a universe of size n. A function $\overline{\boldsymbol{p}} : \{1, 2, \ldots, t\} \to \{0, 1, \ldots, k/t\}$ such that $\sum_{i=1}^{t} p_i = p$, is called* (p, q, t)-compatible. *When $\overline{\boldsymbol{p}}$ is clear from context, for each $i \in \{1, 2, \ldots, t\}$, denote $p_i = \overline{\boldsymbol{p}}(i)$ and $q_i = (k/t) - p_i$.*

A triple $(h, f, \overline{\boldsymbol{p}})$ is called (n, p, q, t, z)-compatible *if $h : U \to \{1, 2, \ldots, z\}$, $f : \{1, 2, \ldots, z\} \to \{1, 2, \ldots, t\}$, and $\overline{\boldsymbol{p}}$ is (p, q, t)-compatible. (The universe U will be clear from context.)*

We begin by defining what is a nice pair.

▶ **Definition 12** (Nice Pair). *Let $n, p, q, t, z \in \mathbb{N}$. Let U be a universe of size n. Let $(h, f, \overline{\mathbf{p}})$ be (n, p, q, t, z)-compatible. A pair (A, B) is* nice *(with respect to $(h, f, \overline{\mathbf{p}})$) if $A \in \binom{U}{p}$ and $B \in \binom{U}{q}$ are disjoint sets, and the following conditions hold.*
1. *The function h is injective when restricted to $A \cup B$.*
2. *For each $i \in \{1, 2, \ldots, t\}$, it holds that $|\{u \in A : f(h(u)) = i\}| = p_i$ and $|\{u \in B : f(h(u)) = i\}| = (k/t) - p_i$.*

Towards the definition of a δ-parsimonious universal family for nice pairs, we first present a weaker definition of this notion where we have a triple $(h, f, \overline{\mathbf{p}})$ at hand.

▶ **Definition 13** (Specific δ-Parsimonious Universal Family for Nice Pairs). *Let $n, p, q, t, z \in \mathbb{N}$. Let U be a universe of size n. Let $(h, f, \overline{\mathbf{p}})$ be (n, p, q, t, z)-compatible. Let $0 < \delta < 1$. A family $\mathcal{F}$ of sets over $\{1, \ldots, z\}$ is a* δ-parsimonious $(h, f, \overline{\mathbf{p}})$-universal family (for nice pairs) *if there exists $T = T(h, f, \overline{\mathbf{p}}, \delta) > 0$ such that for every nice pair (A, B), it holds that $(1 - \delta) \cdot T \leq |\mathcal{F}[h(A), h(B)]| \leq (1 + \delta) \cdot T$.*

Before we show how to extend Definition 13 to the notion useful for applications, we argue that small δ-parsimonious $(h, f, \overline{\mathbf{p}})$-universal families can be computed "efficiently".

▶ **Lemma 14.** *Let $p, q, t, z \in \mathbb{N}$, and denote $k = p + q$ and $s = k/t$. Let $(h, f, \overline{\mathbf{p}})$ be (n, p, q, t, z)-compatible. Let $0 < \delta < 1$. A δ-parsimonious $(h, f, \overline{\mathbf{p}})$-universal family $\mathcal{F}$ of sets over $\{1, \ldots, z\}$ of size $\ell = \mathcal{O}\left(\binom{k}{p} \cdot (k \cdot \log z \cdot \frac{\mathcal{O}(1)}{\delta})^{2t}\right)$ can be computed in time $\ell \cdot z^{s+1} s^{\mathcal{O}(1)} t$. In particular, the sets in $\mathcal{F}$ can be enumerated with delay $z^{s+1} s^{\mathcal{O}(1)} t$.*

Towards the definition of our general construction, we need to present the definitions of a balanced splitter and a balanced hash family. Constructions of such a splitter and a family were given by Alon and Gutner [4, 3].

▶ **Definition 15** (Definition 2.2 [4]). *Suppose that $1 \leq \ell \leq k \leq n$ and $0 < \epsilon < 1$, and let H be a family of functions from $\{1, \ldots, n\}$ to $\{1, \ldots, \ell\}$. For a set $S \in \binom{\{1, \ldots, n\}}{k}$, let $\mathtt{split}_H(S)$ denote the number of functions $h \in H$ that split H into equal size parts, that is, $|h^{-1}(i) \cap S| = k/\ell$. Then, H is an* ϵ-balanced (n, k, ℓ)-splitter *if there exists $T = T(n, k, \ell, \epsilon) > 0$ such that for every set $S \in \binom{\{1, \ldots, n\}}{k}$, we have $(1 - \epsilon)T \leq \mathtt{split}_H(S) \leq (1 - \epsilon)T$.*

▶ **Definition 16** (Definition 2.1 [4]). *Suppose that $1 \leq k \leq \ell \leq n$ and $0 < \epsilon < 1$. A family H of functions from $\{1, \ldots, n\}$ to $\{1, \ldots, \ell\}$ is an* (ϵ, k)-balanced family of hash functions *if there exists $T = T(n, k, \ell, \epsilon) > 0$ such that for every set $S \in \binom{\{1, \ldots, n\}}{k}$, the number of functions in H that are injective when restricted to S is between $(1 - \epsilon)T$ and $(1 + \epsilon)T$.*

We are now ready to define our general derandomization tool.

▶ **Definition 17** ((General) δ-Parsimonious Universal Family for Nice Pairs). *Let $n, p, q \in \mathbb{N}$ and $0 < \delta < 1$, and denote $k = p + q$, $z = \frac{2k^2}{\epsilon}$, $t = \sqrt{k}$, $s = k/t = \sqrt{k}$, and $\epsilon = \delta/3$. Let U be a universe of size n. A* δ-parsimonious (n, p, q)-universal tuple (for nice pairs) *is a tuple $(H, S, \{\mathcal{F}^{h,f,\overline{\mathbf{p}}}\}|_{h \in H, f \in S, \overline{\mathbf{p}}})$[4] that satisfies the following conditions.*
- *H is an (ϵ, k)-balanced family of hash functions from $\{1, \ldots, n\}$ to $\{1, \ldots, z\}$ (with correction factor T_H).*
- *S is an ϵ-balanced (z, k, t)-splitter (with correction factor T_S).*
- *For every hash function $h \in H$, splitter $f \in S$ and (p, q, t)-compatible function $\overline{\mathbf{p}}$, it holds that $\mathcal{F}^{h,f,\overline{\mathbf{p}}}$ is a δ-parsimonious $(h, f, \overline{\mathbf{p}})$-universal family (with correction factor $T_{\overline{\mathbf{p}}}$).*

[4] The enumeration is over every (p, q, t)-compatible $\overline{\mathbf{p}}$.

By *enumerating the quadruples of* $(H,S,\{\mathcal{F}^{h,f,\overline{\mathbf{p}}}\}|_{h\in H,f\in S,\overline{\mathbf{p}}})$, we refer to the enumeration of every quadruple $(h,f,\overline{\mathbf{p}},F)$ such that $h\in H$, $f\in S$ and $F\in\mathcal{F}^{h,f,\overline{\mathbf{p}}}$. We remark that below, for the sake of brevity, when we write k,z,t,s,ϵ,T_H,T_S and $T_{\overline{\mathbf{p}}}$, we refer to the notations given in Definition 17. Let us now state our construction.

▶ **Theorem 18.** *Let $n,p,q\in\mathbb{N}$ and $0<\delta<1$. Denote $k=p+q$. Let U be a universe of size n. A δ-parsimonious (n,p,q)-universal tuple $(H,S,\{\mathcal{F}^{h,f,\overline{\mathbf{p}}}\}|_{h\in H,f\in S,\overline{\mathbf{p}}})$ with ℓ quadruples can be computed in time $\frac{k^{\mathcal{O}(1)}n\log n}{\delta^{\mathcal{O}(1)}}+\ell\cdot\Delta$. In particular, after preprocessing time $\frac{k^{\mathcal{O}(1)}n\log n}{\delta^{\mathcal{O}(1)}}$, the quadruples of $(H,S,\{\mathcal{F}^{h,f,\overline{\mathbf{p}}}\}|_{h\in H,f\in S,\overline{\mathbf{p}}})$ can be enumerated with delay Δ. Here,*

$$\begin{aligned}\ell &= \binom{k}{p}\cdot 2^{\mathcal{O}(\sqrt{k}(\log^2 k+\log^2\frac{1}{\delta}))}\cdot\log n, \text{ and}\\ \Delta &= 2^{\mathcal{O}(\sqrt{k}(\log k+\log\frac{1}{\delta}))}.\end{aligned}$$

In order to state the property of a δ-parsimonious (n,p,q)-universal tuple that makes it useful for applications, we need one last definition.

▶ **Definition 19.** *Let $n,p,q\in\mathbb{N}$ and $0<\delta<1$. Let U be a universe of size n. Furthermore, let $(H,S,\{\mathcal{F}^{h,f,\overline{\mathbf{p}}}\}|_{h\in H,f\in S,\overline{\mathbf{p}}})$ be a δ-parsimonious (n,p,q)-universal tuple. Finally, let $A\in\binom{U}{p}$ and $B\in\binom{U}{q}$ be disjoint sets. We say that the pair (A,B)* fits *a quadruple $(h,f,\overline{\mathbf{p}},F)$ of $(H,S,\{\mathcal{F}^{h,f,\overline{\mathbf{p}}}\}|_{h\in H,f\in S,\overline{\mathbf{p}}})$ if (A,B) is nice with respect to $(h,f,\overline{\mathbf{p}})$, and $h(A)\subseteq F$ and $f\cap h(B)=\emptyset$.*

Finally, we state the promised property.

▶ **Lemma 20.** *Let $n,p,q\in\mathbb{N}$ and $0<\delta<1$. Let U be a universe of size n. Furthermore, let $(H,S,\{\mathcal{F}^{h,f,\overline{\mathbf{p}}}\}|_{h\in H,f\in S,\overline{\mathbf{p}}})$ be a δ-parsimonious (n,p,q)-universal tuple. Then, there exist $T=T(n,p,q,\delta)>0$ and for every $\overline{\mathbf{p}}$ that is (p,q,t)-compatible, $T_{\overline{\mathbf{p}}}=T_{\overline{\mathbf{p}}}(n,p,q,\delta)>0$, such that for any $A\in\binom{U}{p}$ and $B\in\binom{U}{q}$ that are disjoint, the following conditions hold.*

1. *The number of triples $(h,f,\overline{\mathbf{p}})$ with respect to whom (A,B) is nice, where $h\in H$, $f\in S$ and $\overline{\mathbf{p}}$ is (p,q,t)-compatible, is between $(1-\delta)T$ and $(1+\delta)T$.*
2. *For any triple $(h,f,\overline{\mathbf{p}})$ with respect to whom (A,B) is nice, where $h\in H$, $f\in S$ and $\overline{\mathbf{p}}$ is (p,q,t)-compatible, the number of quadruples $(h,f,\overline{\mathbf{p}},F)$ of $(H,S,\{\mathcal{F}^{h,f,\overline{\mathbf{p}}}\}|_{h\in H,f\in S,\overline{\mathbf{p}}})$ that fit (A,B) is between $(1-\delta)T_{\overline{\mathbf{p}}}$ and $(1+\delta)T_{\overline{\mathbf{p}}}$.*

3.5 Deterministic FPT-AS for #k-Path

Our deterministic FPT-AS builds upon the scheme of our second randomized FPT-AS, but it is more technical. Due to space constraints, the full details of the description of the algorithm and its proof of correctness is omitted. Here, we only discuss the main idea that underlies the design of this algorithm. Like our previous algorithm, this algorithm (denoted by $\mathcal{A}$) is recursive. However, in addition to G', k' and α', every call to $\mathcal{A}$ is also given two tuples $\mathcal{R}$ and $\mathcal{W}$. The number of elements in $\mathcal{R}$ and $\mathcal{W}$ equals the depth d of the current recursive call in the recursion tree.

Roughly speaking, every element in $\mathcal{R}$ is a quadruple $(h_i,f_i,\overline{\mathbf{p}}_i,\sigma_i)$ where *(i)* the triple $(h_i,f_i,\overline{\mathbf{p}}_i)$ corresponds to the interpretation preceding Definition 11, and *(ii)* $\sigma_i\in\{\texttt{left},\texttt{right}\}$ indicates whether we should count paths that consist of $\overline{\mathbf{p}}_i(j)$ (in case $\sigma_i=\texttt{left}$) or $s_i-\overline{\mathbf{p}}_i(j)$ (in case $\sigma_i=\texttt{right}$) vertices of the j-th part of the reduced universe split by f_i. Thus, we "keep track" of all triples considered along the current recursion branch. The reason why we have to store this information is to ensure that, in the current recursive call, we only count paths P whose vertex set has the following property:

when we will return to the i-th recursive call, the partition (A, B) of $V(P)$ where A consists of the first $\widehat{k}$ vertices of P (for a certain $\widehat{k} \in \{1, 2, \ldots, k\}$ that depends on the location of this i-th call in the recursion tree) is nice with respect to $(h_i, f_i, \overline{\mathbf{p}}_i)$, see Definition 12. This simple (though perhaps slightly tedious) bookkeeping sidesteps the fact that Lemma 20 only suits nice pairs.

The tuple $\mathcal{W}$ is meant to keep track of how many vertices the paths that we currently count have used "so far" from the j-th part of the universe split by f_i for every choice of i and j. For this purpose, $\mathcal{W}$ is defined to have the form $(\overline{\mathbf{w}}_1, \overline{\mathbf{w}}_2, \ldots, \overline{\mathbf{w}}_d)$ such that for each $i \in \{1, 2, \ldots, d\}$, the following condition holds: For each $j \in \{1, 2, \ldots, t_i\}$, if $\sigma_i = \texttt{left}$ then $\overline{\mathbf{w}}_i(j) \leq \overline{\mathbf{p}}_i(j)$, and otherwise $\overline{\mathbf{w}}_i(j) \leq s_i - \overline{\mathbf{p}}_i(j)$. Here, $s_i = \sqrt{(k/2^i)}$ is the number of vertices the paths that we currently count should use (in total) from each part split by f_i.

Accordingly, the objective of a call $\mathcal{A}(G', k', \alpha', \mathcal{R}, \mathcal{W})$ is to output an assignment $\alpha : V(G') \to \mathbb{N}_0$ with the following property: For each vertex $v \in V(G')$, it holds that $\alpha(v)$ approximates $\sum_{\substack{\{p,q\} \in E(G) \\ \text{s.t. } p \notin V(G'), q \in V(G')}} \alpha'(p) \cdot |\mathcal{P}_{q,v}^{G',k',\mathcal{R},\mathcal{W}}|$. Roughly speaking, $\mathcal{P}_{q,v}^{G',k',\mathcal{R},\mathcal{W}}$ is the collection of all k'-paths in G' with endpoints q and v that "comply" with the constraints imposed by $\mathcal{R}$ and $\mathcal{W}$. (Due to space constraints, the formal definition is omitted.)

We conclude this section with the formal statement of our main result.

▶ **Theorem 21.** *There is a deterministic $4^{k+\mathcal{O}(\sqrt{k}(\log^2 k + \log^2 \frac{1}{\epsilon}))} m \log n$-time polynomial-space algorithm that, given a graph G, a positive integer k and an accuracy value $0 < \epsilon < 1$, outputs a number y that satisfies $(1-\epsilon)x \leq y/2 \leq (1+\epsilon)x$ where x is the number of k-paths in G. In particular, if $\frac{1}{\epsilon} = 2^{o(k^{\frac{1}{4}})}$, then the running time is $4^{k+o(k)} m \log n$.*

Due to space constraints, the discussion on extensions and other applications is omitted.

References

1. Randomization in Parameterized Complexity. `www.dagstuhl.de/de/programm/kalender/semhp/?semnr=17041`.
2. Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and Süleyman Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. In *Proceedings 16th International Conference on Intelligent Systems for Molecular Biology (ISMB), Toronto, Canada, July 19-23, 2008*, pages 241–249, 2008. `doi:10.1093/bioinformatics/btn163`.
3. Noga Alon and Shai Gutner. Balanced Hashing, Color Coding and Approximate Counting. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, pages 1–16, 2009. `doi:10.1007/978-3-642-11269-0_1`.
4. Noga Alon and Shai Gutner. Balanced families of perfect hash functions and their applications. *ACM Trans. Algorithms*, 6(3):54:1–54:12, 2010. `doi:10.1145/1798596.1798607`.
5. Noga Alon, Raphael Yuster, and Uri Zwick. Color-Coding. *J. ACM*, 42(4):844–856, 1995. `doi:10.1145/210332.210337`.
6. Vikraman Arvind and Venkatesh Raman. Approximation Algorithms for Some Parameterized Counting Problems. In *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, pages 453–464, 2002. `doi:10.1007/3-540-36136-7_40`.
7. André Berger, László Kozma, Matthias Mnich, and Roland Vincze. A time- and space-optimal algorithm for the many-visits TSP. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1770–1782, 2019. `doi:10.1137/1.9781611975482.106`.

8 Andreas Björklund. Determinant Sums for Undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

9 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting Paths and Packings in Halves. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, pages 578–586, 2009. doi:10.1007/978-3-642-04128-0_52.

10 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.

11 Andreas Björklund, Vikram Kamat, Lukasz Kowalik, and Meirav Zehavi. Spotting Trees with Few Leaves. *SIAM J. Discrete Math.*, 31(2):687–713, 2017. doi:10.1137/15M1048975.

12 Andreas Björklund, Petteri Kaski, and Lukasz Kowalik. Counting Thin Subgraphs via Packings Faster than Meet-in-the-Middle Time. *ACM Trans. Algorithms*, 13(4):48:1–48:26, 2017. doi:10.1145/3125500.

13 Cornelius Brand, Holger Dell, and Thore Husfeldt. Extensor-coding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 151–164, 2018. doi:10.1145/3188745.3188902.

14 J. Chen, J. Kneis, S. Lu, D. Mölle, S. Richter, P. Rossmanith, S. Sze, and F. Zhang. Randomized Divide-and-Conquer: Improved Path, Matching, and Packing Algorithms. *SIAM Journal on Computing*, 38(6):2526–2547, 2009.

15 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms Are a Good Basis for Counting Small Subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 210–223, New York, NY, USA, 2017. ACM. doi:10.1145/3055399.3055502.

16 Radu Curticapean and Dániel Marx. Complexity of Counting Subgraphs: Only the Boundedness of the Vertex-Cover Number Counts. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 130–139, 2014. doi:10.1109/FOCS.2014.22.

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

18 Banu Dost, Tomer Shlomi, Nitin Gupta, Eytan Ruppin, Vineet Bafna, and Roded Sharan. QNet: A Tool for Querying Protein Interaction Networks. *Journal of Computational Biology*, 15(7):913–925, 2008. doi:10.1089/cmb.2007.0172.

19 Jörg Flum and Martin Grohe. The Parameterized Complexity of Counting Problems. *SIAM J. Comput.*, 33(4):892–922, 2004.

20 Fedor V. Fomin, Petteri Kaski, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Parameterized Single-Exponential Time Polynomial Space Algorithm for Steiner Tree. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 494–505, 2015. doi:10.1007/978-3-662-47672-7_40.

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

22 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2018.

23 Gregory Z. Gutin, Felix Reidl, Magnus Wahlström, and Meirav Zehavi. Designing deterministic polynomial-space algorithms by color-coding multivariate polynomials. *J. Comput. Syst. Sci.*, 95:69–85, 2018. doi:10.1016/j.jcss.2018.01.004.

24 Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm Engineering for Color-Coding with Applications to Signaling Pathway Detection. *Algorithmica*, 52(2):114–132, 2008. doi:10.1007/s00453-007-9008-7.

25 Stasys Jukna. *Extremal Combinatorics: With Applications in Computer Science*. Springer Publishing Company, Incorporated, 1st edition, 2010.

26 Ioannis Koutis. Faster Algebraic Algorithms for Path and Packing Problems. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 575–586, 2008. doi:10.1007/978-3-540-70575-8_47.

27 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, 2016. doi:10.1145/2742544.

28 Ioannis Koutis and Ryan Williams. LIMITS and applications of group algebras for parameterized problems. *ACM Trans. Algorithms*, 12(3):31:1–31:18, 2016. doi:10.1145/2885499.

29 Daniel Lokshtanov, Matthias Mnich, and Saket Saurabh. Planar k-Path in Subexponential Time and Polynomial Space. In *Graph-Theoretic Concepts in Computer Science - 37th International Workshop, WG 2011, Teplá Monastery, Czech Republic, June 21-24, 2011. Revised Papers*, pages 262–270, 2011. doi:10.1007/978-3-642-25870-1_24.

30 Daniel Lokshtanov and Jesper Nederlof. Saving space by algebraization. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 321–330, 2010. doi:10.1145/1806689.1806735.

31 R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, 2002. doi:10.1126/science.298.5594.824.

32 Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

33 Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and Near-Optimal Derandomization. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 182–191, 1995. doi:10.1109/SFCS.1995.492475.

34 Jacob Scott, Trey Ideker, Richard M. Karp, and Roded Sharan. Efficient Algorithms for Detecting Signaling Pathways in Protein Interaction Networks. *Journal of Computational Biology*, 13(2):133–144, 2006. doi:10.1089/cmb.2006.13.133.

35 Hadas Shachnai and Meirav Zehavi. Representative families: A unified tradeoff-based approach. *J. Comput. Syst. Sci.*, 82(3):488–502, 2016. doi:10.1016/j.jcss.2015.11.008.

36 Roded Sharan and Trey Ideker. Modeling cellular machinery through biological network comparison. Nat. Biotechnol. 24, 427-433. *Nature biotechnology*, 24:427–33, May 2006.

37 Tomer Shlomi, Daniel Segal, Eytan Ruppin, and Roded Sharan. QPath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics*, 7:199, 2006. doi:10.1186/1471-2105-7-199.

38 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009. doi:10.1016/j.ipl.2008.11.004.

39 Virginia Vassilevska Williams and Ryan Williams. Finding, Minimizing, and Counting Weighted Subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. doi:10.1137/09076619X.

40 Meirav Zehavi. Mixing Color Coding-Related Techniques. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 1037–1049, 2015. doi:10.1007/978-3-662-48350-3_86.