APPROXIMATELY COUNTING AND SAMPLING SMALL WITNESSES USING A COLOURFUL DECISION ORACLE

HOLGER DELL*, JOHN LAPINSKAS†, AND KITTY MEEKS‡

ABSTRACT. In this paper, we prove "black box" results for turning algorithms which decide whether or not a witness exists into algorithms to approximately count the number of witnesses, or to sample from the set of witnesses approximately uniformly, with essentially the same running time. We do so by extending the framework of Dell and Lapinskas (STOC 2018), which covers decision problems that can be expressed as edge detection in bipartite graphs given limited oracle access; our framework covers problems which can be expressed as edge detection in arbitrary k-hypergraphs given limited oracle access. (Simulating this oracle generally corresponds to invoking a decision algorithm.) This includes many key problems in both the fine-grained setting (such as k-SUM, k-OV and weighted k-Clique) and the parameterised setting (such as induced subgraphs of size k or weight-k solutions to CSPs). From an algorithmic standpoint, our results will make the development of new approximate counting algorithms substantially easier; indeed, it already yields a new state-of-the-art algorithm for approximately counting graph motifs, improving on Jerrum and Meeks (JCSS 2015) unless the input graph is very dense and the desired motif very small. Our k-hypergraph reduction framework generalises and strengthens results in the graph oracle literature due to Beame et al. (ITCS 2018) and Bhattacharya et al. (CoRR abs/1808.00691).

* IT UNIVERSITY OF COPENHAGEN, COPENHAGEN, DENMARK
† UNIVERSITY OF OXFORD, OXFORD, UK
‡ UNIVERSITY OF GLASGOW, GLASGOW, UK
E-mail addresses: hold@itu.dk, john.lapinskas@cs.ox.ac.uk, Kitty.Meeks@glasgow.ac.uk.
Date: July 11, 2019.
The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein. The research was also supported by a Royal Society of Edinburgh Personal Research Fellowship, funded by the Scottish Government.

1. INTRODUCTION

Many decision problems reduce to the question: Does a witness exist? Such problems admit a natural counting version: How many witnesses exist? For example, one may ask whether a bipartite graph contains a perfect matching, or how many perfect matchings it contains. As one might expect, the counting version is never easier than the decision version, and is often substantially harder; for example, deciding whether a bipartite graph contains a perfect matching is easy, and counting the number of such matchings is #P-complete [42]. However, even when the counting version of a problem is hard, it is often easy to approximate well. For example, Jerrum, Sinclair and Vigoda [32] gave a polynomial-time approximation algorithm for the number of perfect matchings in a bipartite graph. The study of approximate counting has seen amazing progress over the last two decades, particularly in the realm of trichotomy results for general problem frameworks such as constraint satisfaction problems, and is now a major field of study in its own right [17, 18, 24, 27, 28]. In this paper, we explore the question of when approximating the counting version of a problem is not merely fast, but essentially as fast as solving the decision version.

We first recall the standard notion of approximation in the field: For all real $x, y > 0$ and $0 < \varepsilon < 1$, we say that x is an *ε-approximation* to y if $|x - y| < \varepsilon y$. Note in particular that any ε-approximation to zero is itself zero, so computing an ε-approximation to N is always at least as hard as deciding whether $N > 0$ holds. For example, it is at least as hard to approximately count the number of satisfying assignments of a CNF formula (i.e. to ε-approximate #SAT) as it is to decide whether it is satisfiable at all (i.e. to solve SAT).

Perhaps surprisingly, in many cases, the converse is also true. For example, Valiant and Vazirani [43] proved that any polynomial-time algorithm to decide SAT can be bootstrapped into a polynomial-time ε-approximation algorithm for #SAT, or, more formally, that a size-n instance of any problem in #P can be ε-approximated in time $\mathrm{poly}(n, \varepsilon^{-1})$ using an NP-oracle. A similar result holds in the parameterised setting, where Müller [40] proved that a size-n instance of any problem in #W[i] with parameter k can be ε-approximated in time $g(k) \cdot \mathrm{poly}(n, \varepsilon^{-1})$ using a W[i]-oracle for some computable function $g\colon \mathbb{N} \to \mathbb{N}$. Another such result holds in the subexponential setting, where Dell and Lapinskas [14] proved that the (randomised) Exponential Time Hypothesis is equivalent to the statement: There is no ε-approximation algorithm for #3-SAT which runs on an n-variable instance in time $\varepsilon^{-2} 2^{o(n)}$.

We now consider the fine-grained setting, which is the focus of this paper. Here, we are concerned with the exact running time of an algorithm, rather than broad categories such as polynomial time, FPT time or subexponential time. The above reductions all introduce significant overhead, so they are not fine-grained. Here only one general result is known, again due to Dell and Lapinskas [14]. Informally, if the decision problem reduces "naturally" to deciding whether an n-vertex bipartite graph contains an edge, then any algorithm for the decision version can be bootstrapped into an ε-approximation algorithm for the counting version with only $\mathcal{O}(\varepsilon^{-2}\mathrm{polylog}(n))$ overhead. (See Section 1.1 for more details.)

The reduction of [14] is general enough to cover core problems in fine-grained complexity such as ORTHOGONAL VECTORS, 3SUM and NEGATIVE-WEIGHT TRIANGLE, but it is not universal. In this paper, we substantially generalise it to cover any problem which can be "naturally" formulated as deciding whether a k-partite k-hypergraph contains an edge; thus we essentially recover the original result on taking $k = 2$. For any problem which satisfies this property, our result implies that any new decision algorithm will automatically lead to a new approximate counting algorithm whose running time is at most a factor of $\log^{\mathcal{O}(k)} n$ larger. Our framework covers several reduction targets in fine-grained complexity not covered by [14], including k-ORTHOGONAL VECTORS, k-SUM and EXACT-WEIGHT k-CLIQUE, as well as some key problems in parameterised complexity including weight-k CSPs and size-k induced subgraph problems. (Note that the overhead of $\log^{\mathcal{O}(k)} n$ can be re-expressed as $k^{2k} n^{o(1)}$ using a standard trick, so an FPT decision algorithm is transformed into an FPT approximate counting algorithm; see Section 1.3.)

In fact, we get more than fast approximate counting algorithms — we also prove that any problem in this framework has an algorithm for approximately-uniform sampling, again with $\log^{\mathcal{O}(k)} n$ overhead over decision. There is a well-known reduction between the two for self-reducible problems due to Jerrum, Valiant and Vazirani [33], but it does not apply in our setting since it adds polynomial overhead.

In the parameterised setting, our results have interesting implications. Here, the requirement that the hypergraph be k-partite typically corresponds to considering the "colourful" or "multicolour" version of the decision problem, so our result implies that uncoloured approximate counting is essentially equivalent to multicolour decision. We believe that our results motivate considerable further study of the relationship between multicolour parameterised decision problems and their uncoloured counterparts.

Finally, we note that the applications of our results are not just complexity-theoretic in nature, but also algorithmic. They give a "black box" argument that any decision algorithm in our framework, including fast ones, can be converted into an approximate counting or sampling algorithm with minimal overhead. Concretely, we obtain new algorithms for approximately counting and/or sampling zero-weight subgraphs, graph motifs, and satisfying assignments for first-order models, and our framework is sufficiently general that we believe new applications will be forthcoming.

In Section 1.1, we set out our main results in detail as Theorems 1 and 2, and discuss our edge-counting reduction framework (which is of independent interest). We describe the applications of Theorems 1 and 2 to fine-grained complexity in Section 1.2, and their applications to parameterised complexity in Section 1.3.

1.1. The k-hypergraph framework.

Given a k-hypergraph $G = (V, E)$, write $e(G) = |E|$, and let

$$\mathcal{C}(G) := \{(X_1, \dots, X_k) \colon X_1, \dots, X_k \text{ are disjoint subsets of } V\}.$$

We define the *coloured independence oracle* of G to be the function $\mathrm{cIND}_G \colon \mathcal{C}(G) \to \{0, 1\}$ such that $\mathrm{cIND}_G(X_1, \dots, X_k) = 1$ if $G[X_1, \dots, X_k]$ has no edges, and $\mathrm{cIND}_G(X_1, \dots, X_k) = 0$ otherwise. Informally, we think of elements of $\mathcal{C}(G)$ as representing k-colourings of induced subgraphs of G, with X_i being the i'th colour class; thus given a vertex colouring of an induced subgraph of G, the coloured independence oracle outputs 1 if and only if no colourful edge is present. We consider a computation model where the algorithm is given access to V and k, but can only access E via cIND_G. We say that such an algorithm has *coloured oracle access* to G, and for legibility we write it to have G as an input. Our main result is as follows.

Theorem 1. *There is a randomised algorithm* $\texttt{Count}(G, \varepsilon, \delta)$ *with the following behaviour. Suppose G is an n-vertex k-hypergraph, and that* `Count` *has coloured oracle access to G. Suppose ε and δ are rational with $0 < \varepsilon, \delta < 1$. Then, writing $T = \log(1/\delta)\varepsilon^{-2}k^{6k}\log^{4k+7} n$: in time $\mathcal{O}(nT)$, and using at most $\mathcal{O}(T)$ queries to* cIND_G, $\texttt{Count}(G, \varepsilon, \delta)$ *outputs a rational number $\hat{e}$. With probability at least $1 - \delta$, we have $\hat{e} \in (1 \pm \varepsilon)e(G)$.*

As an example of how Theorem 1 applies to approximate counting problems, consider the problem #k-CLIQUE of counting the number of cliques in an n-vertex graph H of size k. We take G to be the k-hypergraph on vertex set $V(H)$ whose hyperedges are precisely those size-k sets which span cliques in G. Thus ε-approximating the number of k-cliques in H corresponds to ε-approximating the number of hyperedges in G. We may use a decision algorithm for k-Clique with running time $f(n, k)$ to evaluate cIND_G in time $f(n, k)$, by applying it to an appropriate subgraph of G (in which we delete all edges within each colour class X_i). Thus Theorem 1 gives us an algorithm for ε-approximating the number of k-cliques in H in time $\mathcal{O}(nT + Tf(n, k))$. Any decision algorithm for k-Clique must read a constant proportion of its input, so we have $f(n, k) = \Omega(n)$ and our overall running time is $\mathcal{O}(Tf(n, k))$. It follows that any decision algorithm for k-clique yields an ε-approximation algorithm for #k-Clique with overhead only $T = \varepsilon^{-2}(k \log n)^{\mathcal{O}(k)}$.

The polynomial dependence on ε in Theorem 1 is not surprising, as by taking $\varepsilon < 1/2n^k$ and rounding we can obtain the number of edges of G exactly. Thus if the dependence on ε were subpolynomial, Theorem 1 would essentially imply a fine-grained reduction from exact counting to decision. This is impossible under SETH in our setting; see [14, Theorem 3] for a more detailed discussion.

We extend Theorem 1 to approximately-uniform sampling as follows.

Theorem 2. *There is a randomised algorithm* $\texttt{Sample}(G, \varepsilon)$ *which, given a rational number ε with $0 < \varepsilon < 1$ and coloured oracle access to an n-vertex k-hypergraph G containing at least one edge, outputs either a random edge $f \in E(G)$ or* `Fail`*. For all $f \in E(G)$,* $\texttt{Sample}(G, \varepsilon)$ *outputs f with*

probability $(1 \pm \varepsilon)/e(G)$*; in particular, it outputs* `Fail` *with probability at most* ε*. Moreover, writing* $T = \varepsilon^{-2}k^{7k}\log^{4k+11} n$*,* $\texttt{Sample}(G, \varepsilon)$ *runs in time* $\mathcal{O}(nT)$ *and uses at most* $\mathcal{O}(T)$ *queries to* cIND_G*.*

We call the output of this algorithm an ε*-approximate sample*. Note that there is a standard trick using rejection sampling which, given an algorithm of the above form, replaces the ε^{-2} factor in the running time by a $\mathrm{polylog}(\varepsilon^{-1})$ factor; see [33]. Unfortunately, it does not apply to Theorem 2, as we do not have a fast way to compute the true distribution of `Sample`'s output.

By the same argument as above, Theorem 2 may be used to sample a size-k clique from a distribution with total variation distance at most ε from uniformity with overhead only $T = \varepsilon^{-2}(k \log n)^{\mathcal{O}(k)}$ over decision. (We also note that it is easy to extend Theorems 1 and 2 to cover the case where the original decision algorithm is randomised, at the cost of an extra factor of $k \log n$ in the number of oracle uses; we discuss this further in the full version.)

Theorems 1 and 2 are also of independent interest, generalising known results in the graph oracle literature. Our colourful independence oracles are a natural generalisation of the bipartite independent set (BIS) oracles of Beame et al. [6] to a hypergraph setting, and when $k = 2$ the two notions coincide. Their main result [6, Theorem 4.9] says that given BIS oracle access to an n-vertex graph G, one can ε-approximate the number of edges of G using $\mathcal{O}(\varepsilon^{-4}\log^{14} n)$ BIS queries (which they take as their measure of running time). The $k = 2$ case of Theorem 1 gives a total of $\mathcal{O}(\varepsilon^{-2}\log^{19} n)$ queries used, improving their running time for most values of ε, and Theorem 2 extends their algorithm to approximately-uniform sampling.

When $k = 3$, our colourful independence oracles are similar to the tripartite independent set (TIS) oracles of Bhattacharya et al. [8]. (These oracles ask whether a 3-coloured graph H contains a colourful triangle, rather than whether a 3-coloured 3-hypergraph G contains a colourful edge. But if G is taken to be the 3-hypergraph whose edges are the triangles of H, then the two notions coincide exactly.) Their main result, Theorem 1, says that given TIS oracle access to an n-vertex graph G of maximum degree at most d, one can ε-approximate the number of triangles in G using at most $\mathcal{O}(\varepsilon^{-12}d^{12}\log^{25} n)$ TIS queries. Our Theorem 1 gives an algorithm which requires only $\mathcal{O}(\varepsilon^{-2}\log^{22} n)$ TIS queries, with no dependence on d, and which also generalises to approximately counting k-cliques for all fixed k. Again, Theorem 2 extends the result to approximately-uniform sampling.

We note in passing that the main result of [14] doesn't quite fit into this setting, as it also makes unrestricted use of edge existence queries. It resembles a version of Theorem 1 restricted to $k = 2$ and with slightly lower overhead in n.

1.2. **Corollaries in fine-grained complexity.** In [14], fine-grained reductions from approximate counting to decision were shown for the problems ORTHOGONAL VECTORS, 3SUM and NEGATIVE-WEIGHT TRIANGLE (among others). The approximate counting procedure for k-uniform hypergraphs in Theorem 1 allows us to generalize these reductions to k-OV, k-SUM, ZERO-WEIGHT k-CLIQUE, and other subgraph isomorphism problems. They also apply to model checking of first-order formulas with k variables. In each case, Theorem 2 yields a corresponding result for approximate sampling of witnesses.

1.2.1. *First-order Formulas on Sparse Structures and* k *Orthogonal Vectors.* We consider first-order formulas φ, that is, formulas of the form: $Q_1 x_{\ell+1} Q_2 x_{\ell+2} \ldots Q_k x_k \,.\, \psi(x_1, \ldots, x_k)$. The variables $x_1, \ldots, x_\ell$ are the free variables of φ, each Q_i is a quantifier from $\{\exists, \forall\}$, and ψ is a quantifier-free Boolean formula over the variables $x_1, \ldots, x_k$. We consider first-order formulas in prenex-normal form with $\ell \in \{0, \ldots, k\}$ free variables and quantifier-rank at most $k - \ell$; let k-FO denote the set of all such formulas. The *property testing problem for* k*-FO* is, given a formula and a structure (e.g., the edge relation of a graph), to decide whether the formula is satisfiable in the structure, that is, whether there is an assignment to the free variables that makes the formula true. Correspondingly, the *property counting problem* is to count all satisfying assignments.

Model checking and property testing are important problems in logic and database theory, and have recently been studied in the context of fine-grained complexity [15, 25, 45]: Gao et al. [25] devise an algorithm for the property testing problem for k-FO that runs in time $m^{k-1}/2^{\Theta(\sqrt{\log m})}$, where m is the number

of distinct tuples in the input relations. This improves upon an already slightly non-trivial $\widetilde{\mathcal{O}}(m^{k-1})$ algorithm.[1] By using this improved decision algorithm as a black box, we obtain new algorithms for approximate counting (via Theorem 1) and approximate sampling (via Theorem 2). Note all our approximate counting algorithms work with probability at least $2/3$; this can easily be increased to $1-\delta$ in the usual way, i.e. running them $\mathcal{O}(\log(1/\delta))$ times and taking the median result.

Corollary 3. *Fix $k \in \mathbb{Z}_{\geq 0}$, suppose an instance of property testing for k-FO can be solved in time $T(n,m) = \mathcal{O}((m+n)^k)$, where n is the size of the universe and m is the number of tuples in the structure, and write $\mathcal{S}$ for the set of satisfying assignments. Then there is a randomised algorithm to ε-approximate $|\mathcal{S}|$, or draw an ε-approximate sample from $\mathcal{S}$, in time $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(n,m))$.*

In combination with the algorithm of Gao et al. [25], we can thus ε-approximately sample from the set of satisfying assignments to any k-FO-property in time $\varepsilon^{-2} m^{k-1}/2^{\Theta(\sqrt{\log m})}$. For example, this algorithm can be used to sample an approximately uniformly random solution tuple to a conjunctive query.

The k-ORTHOGONAL VECTORS (k-OV) problem is a specific example of a property testing problem, and has connections to central conjectures in fine-grained complexity theory [1, 25]. The problem asks, given k sets $X_1, \ldots, X_k \subseteq \{0,1\}^D$ of Boolean vectors, whether there exist $x_1 \in X_1, \ldots, x_k \in X_k$ such that $\sum_{j=1}^{D} \prod_{i=1}^{k} x_{ij} = 0$. (The sum and product are the usual arithmetic operations over $\mathbb{Z}$.) When $x_1, \ldots, x_k$ are viewed as representing subsets of $[D]$ in the canonical manner, this condition is equivalent to requiring they have an empty intersection; when $k=2$, it is equivalent to x_1 and x_2 being orthogonal. Any tuple $(x_1, \ldots, x_k)$ satisfying the condition is called a *witness*. Clearly, k-OV can be solved in time $O(N^k D)$ using exhaustive search. Gao et al. [25] stated the Moderate-Dimension k-OV Conjecture, which says that k-OV cannot be solved in time $O(N^{k-\varepsilon}\operatorname{poly}(D))$ time for any $\varepsilon > 0$. We show that any reasonable-sized improvement over exhaustive search carries over to approximate counting and sampling.

Corollary 4. *Fix $k \geq 2$, suppose an N-vector D-dimension instance of k-OV can be solved in time $T(N,D)$, and write W for the set of witnesses. Then there is a randomised algorithm to ε-approximate $|W|$, or draw an ε-approximate sample from W, in time $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(N,D))$.*

Note that such an improvement is already known for 2-OV, which has an $N^{2-1/\mathcal{O}(\log(D/\log N))}$-time algorithm [3], although Chan and Williams [12] already generalised this to an exact counting algorithm.

1.2.2. *k-SUM.* The k-SUM problem has been studied since the 1990s as it arises naturally in the context of computational geometry, see for example [23], and it has become an important problem in fine-grained complexity theory [46]. For all integers $k \geq 3$, the k-SUM problem asks, given a set of integers, whether some k of them sum to zero. Each k-subset of integers that does sum to zero is called a *witness*. While Kane, Lovett, and Moran [34] very recently developed almost linear-size linear decision trees for k-SUM, the fastest known algorithm for this problem still runs in time $\widetilde{\mathcal{O}}(n^{\lceil k/2 \rceil})$, and $n^{o(k)}$ as $k \to \infty$ is ruled out under the exponential-time hypothesis [41]. We prove that any sufficiently non-trivial improvement over the best known decision algorithm carries over to approximate counting and witness sampling.

Corollary 5. *Fix $k \geq 3$, suppose an n-integer instance of k-SUM can be solved in time $T(n)$, and write W for the set of witnesses. Then there is a randomised algorithm to ε-approximate $|W|$, or draw an ε-approximate sample from W, in time $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(n))$.*

1.2.3. EXACT-WEIGHT k-CLIQUE *and Other Subgraph Problems.* Recall that Theorem 1 applies to the problem #k-CLIQUE. This observation generalizes to other subgraph problems as well. We consider weighted graph problems, where we are given a graph G with an edge-weight function $w : E(G) \to \mathbb{Z}$. The weight of a clique X in G is the sum $\sum_e w(e)$ over all edges $e \in E(G)$ with $e \subseteq X$. The EXACT-WEIGHT k-CLIQUE problem is to decide whether there is a k-clique X of weight exactly 0. It has been conjectured [1] that there is no real $\varepsilon > 0$ and integer $k \geq 3$ such that the EXACT-WEIGHT k-CLIQUE problem

[1] The notation $\widetilde{\mathcal{O}}(f(n,m))$ means $f(n,m) \cdot \operatorname{polylog}(n+m)$.

on n-vertex graphs and with edge-weights in $\{-M, \ldots, M\}$ can be solved in time $\mathcal{O}(n^{(1-\varepsilon)k}\mathrm{polylog}(M))$. (For the closely related MIN-WEIGHT k-CLIQUE problem, a subpolynomial-time improvement over the exhaustive search algorithm is known [1, 44, 12], with running time $n^k/\exp(\Omega(\sqrt{\log n}))$.) Theorems 1 and 2 imply that any sufficiently non-trivial improvement on the running time of an EXACT-WEIGHT k-CLIQUE algorithm will carry over to the approximate counting and sampling versions of the problem.

Corollary 6. *Fix $k \geq 3$, suppose an n-vertex m-edge instance of* EXACT-WEIGHT k-CLIQUE *with weights in $[-M, M]$ can be solved in time $T(n, m, M)$, and write $\mathcal{C}$ for the set of zero-weight k-cliques. Then there is a randomised algorithm to ε-approximate $|\mathcal{C}|$, or draw an ε-approximate sample from $\mathcal{C}$, in time $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(n, m, M))$.*

There is a more general version of EXACT-WEIGHT k-CLIQUE which takes as input an edge-weighted d-hypergraph and asks whether it contains a zero-weight k-clique. A similar conjecture exists for this version of the problem [1], and Theorems 1 and 2 yield a result analogous to Corollary 6.

Our framework also applies to subgraphs more general than cliques. The EXACT-WEIGHT-H problem asks, given an edge-weighted graph G, whether there exists a subgraph of G that has weight zero and is isomorphic to H. We say H is a *core* if every homomorphism from H to H is also an automorphism. Cores are a rich class of graphs, including cliques, odd cycles, and (with high probability) any binomial random graph $G(n, p)$ with edge probability $n^{-1/3}\log^2 n < p < 1 - n^{1/3}\log^2 n$ (see [10, Theorem 2]). Corollary 6 generalises to EXACT-WEIGHT-H whenever H is a core. In particular, Abboud and Lewi [2, Corollary 5] prove that EXACT-WEIGHT-H can be solved in time $\widetilde{\mathcal{O}}(n^{\gamma(H)})$, where $\gamma(H) \geq 1$ is a graph parameter that is small whenever H has a balanced separator, so we obtain the following result.

Corollary 7. *Let H be a core, let G be an n-vertex graph, and let $H(G)$ be the set of zero-weight H-subgraphs in G. There is an algorithm to draw an ε-approximate sample from $H(G)$ in time $\widetilde{\mathcal{O}}(\varepsilon^{-2}n^{\gamma(H)})$.*

Our framework also applies to colourful subgraphs. The COLOURFUL-H problem asks, given a graph G and a vertex colouring $c\colon V(G) \to \{1, \ldots, |V(H)|\}$, whether there G contains a colourful copy of H — that is, a subgraph isomorphic to H containing one vertex from each colour class.

Corollary 8. *Let H be a fixed graph, suppose an n-vertex m-edge instance of* COLOURFUL-H *can be solved in time $T(m, n)$, and write $\mathcal{H}$ for the set of colourful H-subgraphs. Then there is a randomised algorithm to ε-approximate $|\mathcal{H}|$, or draw an ε-approximate sample from $\mathcal{H}$, in time $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(m, n))$.*

Daz, Serna, and Thilikos [16] show using dynamic programming that #COLOURFUL-H can be solved exactly in time $\widetilde{\mathcal{O}}(n^{t+1})$, where t is the treewidth of H. Marx [37] asks whether it is possible to detect colourful subgraphs in time $n^{o(t)}$, and proves that $n^{o(t/\log t)}$ is impossible under the exponential-time hypothesis (ETH). Our result shows that any algorithm to detect colourful subgraphs in time $n^{o(t)}$ would essentially also have to approximately count these subgraphs — a more difficult task.

1.3. **Corollaries in parameterised complexity.** When considering approximation algorithms for parameterised counting problems, an "efficient" approximation scheme is an FPTRAS (fixed parameter tractable randomised approximation scheme), as introduced by Arvind and Raman [5]; this is the analogue of an FPRAS in the parameterised setting. An FPTRAS for a parameterised counting problem Π with parameter k is an algorithm that takes an instance I of Π (with $|I| = n$) and a rational number $\varepsilon > 0$, and in time $f(k) \cdot \mathrm{poly}(n, 1/\varepsilon)$ (where f is some computable function) outputs a rational number z such that

$$\mathbb{P}[(1-\varepsilon)\Pi(I) \leq z \leq (1+\varepsilon)\Pi(I)] \geq 2/3.$$

Note that this definition is equivalent to that given in [5] which requires the failure probability to be at most δ, where δ is part of the input; repeating the process above $\mathcal{O}(\log(1/\delta))$ times and returning the median solution allows us to reduce the error probability from $1/3$ to δ.

As mentioned above, a large number of well-studied problems in parameterised complexity fall within our k-hypergraph framework; for standard notions in parameterised (counting) complexity we refer the

reader to [21]. Observe that we can rewrite our overhead of $\log^{\mathcal{O}(k)} n$ in the form $k^{2k}n^{o(1)}$: if $k \leq \log n/(\log\log n)^2$ then $\log^{\mathcal{O}(k)} n = e^{\mathcal{O}(\log n/\log\log n)} = n^{o(1)}$, and if $k \geq \log n/(\log\log n)^2$ then $\log^{\mathcal{O}(k)} n = \mathcal{O}(k^{2k})$. Thus we can consider this to be a "fine-grained FPT overhead".

Theorems 1 and 2 can therefore be applied immediately to any *self-contained k-witness problem* (see [39]); that is, any problem with integer parameter k in which we are interested in the existence of witnesses consisting of k-element subsets of some given universe, and we have the ability to quickly test whether any given k-element set is such a witness. Examples include weight-k solutions to CSPs, size-k solutions to database queries, and sets of k vertices in a (weighted) graph or hypergraph which induce a sub(hyper)graph with specific properties. This last example encompasses many of the best-studied problems in parameterised counting complexity, including the problem #SUB(H, G) (with parameter $|V(H)|$) which asks for the number of subgraphs of G isomorphic to H; the well-studied problems of counting k-vertex paths, cycles and cliques are all special cases. More generally, we can consider the problem #INDUCED SUBGRAPH WITH PROPERTY(Φ) (#ISWP(Φ)), introduced by Jerrum and Meeks [31], for any property Φ.

However, our coloured independence oracle doesn't quite correspond to deciding whether a witness exists: it needs to solve a *multicolour* version of the decision problem. The multicolour decision version of a self-contained k-witness problem takes as input a universe U together with a k-colouring of the elements of U, and asks whether there exists a witness which contains precisely one element of each colour. The following result is immediate from Theorems 1 and 2 on taking the vertex set of the hypergraph to be U, the edges to be the k-witnesses, and simulating the coloured independence oracle by invoking a multicolour decision algorithm.

Theorem 9. *Let Π be a self-contained k-witness decision problem, and suppose that the multicolour version of Π can be solved in time $T(n, k)$ when the universe U has size n. Let $c\colon U \to [k]$ be a colouring, let W be the set of (uncoloured) witnesses of Π, and let W^c be the set of multicolour witnesses of Π with respect to c. Then given U and c, in time $\varepsilon^{-2}k^{2k}n^{o(1)}T(n, k)$, there is a randomised algorithm to ε-approximate $|W|$ or $|W^c|$, or draw an ε-approximate sample from W or W^c.*

Such multicolour problems have been studied before in the literature, including #MISWP(Φ), the multicolour version of #ISWP(Φ); see [38] for a survey of results relating the complexity of multicolour and uncoloured problems in this setting. In many cases, the multicolour decision problem reduces straightforwardly to the original decision problem — for example, if our witnesses are k-vertex cliques in a graph. But this is not true in general; if our witnesses are k-vertex cliques *and* k-vertex independent sets, then the uncoloured decision problem admits a trivial FPT algorithm by Ramsey's theorem [5], but the W[1]-complete problem k-CLIQUE reduces to the multicolour version [38]. In the restricted setting of SUB(H,G), it is straightforward to verify that the multicoloured and uncoloured versions of the problem are equivalent when the graph H is a core[2], but this is not known for general H. In fact, a proof of equivalence would imply the long-standing dichotomy conjecture for the parameterised embedding problem (see [13] for recent progress on this conjecture). We believe that Theorem 9 motivates substantial further research into the complexity relationship between multicoloured problems and their uncoloured counterparts.

One consequence of Theorem 9 is that if MISWP(Φ) admits an FPT decision algorithm, then we obtain FPTRASes for both #MISWP(Φ) and #ISWP(Φ) with roughly the same running time as the original decision algorithm. This generalises a previous result of Meeks [38, Corollaries 4.8 and 4.10] which states that subject to standard complexity-theoretic assumptions, if we restrict our attention to properties Φ that are preserved under adding edges, there is an FPTRAS for the counting problems #MISWP(Φ) and #ISWP(Φ) if and only if there is an FPT decision algorithm for MISWP(Φ). Theorem 9 strengthens this result in two ways. Firstly, we no longer need the restriction that the property is preserved under adding edges, as we can now consider an arbitrary property Φ. Secondly, we demonstrate a close relationship between the running-times for decision and approximate counting, meaning that any improvement in a decision algorithm immediately translates to an improved algorithm for approximate counting.

[2]The reduction appears inside the proof of Lemma 27.

One example where Theorem 9 already gives an improvement (in almost all settings) to the previously best-known algorithm for approximate counting is the GRAPH MOTIF problem, introduced by Lacroix, Fernandes and Sagot [36] in the context of metabolic networks. This problem takes as input an n-vertex m-edge graph with a (not necessarily proper) vertex-colouring, together with a multiset M of colours, and a solution is a subset U of $|M| = k$ vertices such that the subset induced by U is connected and the colour multiset of U is exactly M; M is called a *motif*, and we call U a *motif witness* for M.

There has been substantial progress in recent years on improving the running-time of decision algorithms for GRAPH MOTIF [7, 9, 19, 26, 35], with the fastest randomised algorithm [9] (based on constrained multilinear detection) running in time $\mathcal{O}(2^k k^3 m)$. For the counting version, Guillemot and Sikora [26] addressed the related problem of counting k-vertex sub*trees* of a graph whose vertex set has colour multiset M (which counts motif witnesses U for M weighted by the the number of trees spanned by U). They demonstrated that this problem admits an FPT algorithm for exact counting when M is a set, but is #W[1]-hard otherwise. Subsequently, Jerrum and Meeks [31] addressed the more natural counting analogue of GRAPH MOTIF in which the goal is to count motif witnesses for M without weights. They demonstrated that this problem is #W[1]-hard to solve exactly even if M is a set, but gave an FPTRAS to solve it approximately. By using this FPTRAS together with Theorems 1 and 2, we prove the following.

Corollary 10. *Given an n-vertex instance of* GRAPH MOTIF *with parameter k and $0 < \varepsilon < 1$, there is a randomised algorithm to ε-approximate the number of motif witnesses or to draw an ε-approximate sample from the set of motif witnesses in time $\mathcal{O}(\varepsilon^{-2} k^{8k} m \log^{4k+8} n)$.*

Theorem 9 also generalises a known relationship between the complexity of uncoloured approximate counting and multicolour decision in the special case of SUB(H,G). In this restricted setting, multicolour decision is actually equivalent to multicolour exact counting; there is an FPT algorithm to exactly count the number of multicolour solutions whenever the treewidth of H is bounded by a constant, with essentially the same running time as the best-known decision algorithm [4]. On the other hand, even the multicolour decision problem is W[1]-hard if H is restricted to any class of graphs with unbounded treewidth [38]. Alon et al. [29] essentially give a fine-grained reduction from uncoloured approximate counting to multicolour exact counting, giving an algorithm with running time matching the best-known algorithm for multicolour decision. (Note that their running time is slightly better than that obtained by applying Theorem 9, and that uncoloured exact counting is #W[1]-hard even when H is a path or cycle [22].)

However, in general it is not true that multicolour exact counting is equivalent to multicolour decision — indeed, there are natural examples (such as counting k-vertex subsets that induce connected subgraphs) in which the counting is #W[1]-hard but the decision is FPT [31]. Theorem 9 therefore strengthens [29], in the sense that if a faster multicolour decision algorithm is discovered then the improvement to the running time will immediately be carried over to uncoloured approximate counting, whether or not the new algorithm generalises to exact multicolour counting.

In this specific case, the existing decision algorithm turns out to already give an algorithm for exact counting with the same asymptotic complexity; however, there is no theoretical reason why the constant in the exponent could not be improved, and our results mean that any such improvement in a decision algorithm could immediately be translated to a faster algorithm for approximate counting.

Organisation. In the following section, we set out our notation and quote some standard probabilistic results for future reference. We then prove Theorem 1 in Section 3.2, using a weaker approximation algorithm which we set out in Section 4. We then prove Theorem 2 (using Theorem 1) in Section 5. Finally, we prove our assorted corollaries in Section 6; we emphasise that in general, the proofs in this section are easy and use only standard techniques.

2. PRELIMINARIES

2.1. **Notation.** Let $k \geq 2$ and let $G = (V, E)$ be a k-hypergraph, so that each edge in E has size exactly k. We write $e(G) = |E|$. For all $U \subseteq V$, we write $G[U]$ for the subgraph induced by U. If $X_1, \ldots, X_k \subseteq V(G)$ are disjoint, then we write $G[X_1, \ldots, X_k]$ for the k-partite k-hypergraph on $X_1 \cup \cdots \cup X_k$ whose edge set

is $\{e \in E(G) \colon |e \cap X_i| = 1 \text{ for all } i \in [k]\}$. For all $S \subseteq V$, we write $d_H(S) = |\{e \in E(G) \colon S \subseteq e\}|$ for the degree of S in H. If $S = \{v_1, \ldots, v_{|S|}\}$, then we will sometimes write $d_H(v_1, \ldots, v_{|S|}) = d_H(S)$.

For all positive integers t, we write $[t] = \{1, \ldots, t\}$. We write $\ln$ for the natural logarithm, and $\log$ for the base-2 logarithm. Given real numbers $x, y \geq 0$ and $0 < \varepsilon < 1$, we say that x is an *ε-approximation* to y if $(1-\varepsilon)x < y < (1+\varepsilon)x$, and write $y \in (1 \pm \varepsilon)x$. We extend this notation to other operations in the natural way, so that (for example) $y \in xe^{\pm\varepsilon}/(2 \mp \varepsilon)$ means that $xe^{-\varepsilon}/(2+\varepsilon) \leq y \leq xe^{\varepsilon}/(2-\varepsilon)$.

When stating quantitative bounds on running times of algorithms, we assume the standard randomised word-RAM machine model with logarithmic-sized words; thus given an input of size N, we can perform arithmetic operations on $\mathcal{O}(\log N)$-bit words and generate uniformly random $\mathcal{O}(\log N)$-bit words in $\mathcal{O}(1)$ time.

Recall the definitions of $\mathcal{C}(G)$ and the coloured independence oracle of G, and coloured oracle access from Section 1.1. Note that for all $X \subseteq V(G)$, $\mathrm{cIND}_{G[X]}$ is a restriction of cIND_G. Thus an algorithm with coloured oracle access to G can safely call a subroutine that requires coloured oracle access to $G[X]$.

2.2. **Probabilistic results.** We use some standard results from probability theory, which we collate here for reference. The following lemma is commonly known as Hoeffding's inequality.

Lemma 11 ([11, Theorem 2.8])**.** *Let $X_1, \ldots, X_m$ be independent real random variables, and suppose there exist $a_1, \ldots, a_m, b_1, \ldots, b_m \in \mathbb{R}$ be such that $X_i \in [a_i, b_i]$ with probability 1. Let $X = \sum_{i=1}^m X_i$. Then for all $t \geq 0$, we have*

$$\mathbb{P}\big(|X - \mathbb{E}(X)| \geq t\big) \leq 2e^{-2t^2/\sum_{i=1}^m (b_i - a_i)^2}. \qquad \square$$

The next lemma is a form of Bernstein's inequality.

Lemma 12. *Let $X_1, \ldots, X_k$ be independent real random variables. Suppose there exist ν and M such that with probability 1, $\sum_i \mathbb{E}(X_i^2) \leq \nu$ and $|X_i| \leq M$ for all $i \in [k]$. Let $X = \sum_{i=1}^k X_i$. Then for all $z \geq 0$, we have*

$$\mathbb{P}\big(|X - \mathbb{E}(X)| \geq z\big) \leq 2\exp\Big(-\frac{3z^2}{6\nu + 2Mz}\Big).$$

Proof. Apply [11, Corollary 2.11] to both X and $-X$, taking $c = M/3$ and $t = z$, then apply a union bound. $\square$

The next lemma collates two standard Chernoff bounds.

Lemma 13 ([30, Corollaries 2.3-2.4])**.** *Suppose X is a binomial or hypergeometric random variable with mean μ. Then:*

(i) for all $0 < \varepsilon \leq 3/2$, $\mathbb{P}(|X - \mu| \geq \varepsilon\mu) \leq 2e^{-\varepsilon^2\mu/3}$;
(ii) for all $t \geq 7\mu$, $\mathbb{P}(X \geq t) \leq e^{-t}$. $\square$

Our final lemma is a standard algebraic bound.

Lemma 14. *For all positive integers N and k with $N \geq 2k^2$, we have $\binom{2N-k}{N-k}\big/\binom{2N}{N} \geq 2^{-k-1}$.*

Proof. We have

$$\begin{aligned}\binom{2N-k}{N-k}\Big/\binom{2N}{N} &= \frac{(2N-k)!N!}{(2N)!(N-k)!} = \prod_{i=0}^{k-1}(N-i)\Big/\prod_{j=0}^{k-1}(2N-j) \\ &\geq \Big(\frac{N-k+1}{2N-k+1}\Big)^k = \Big(\frac{1}{2} - \frac{k-1}{2(2N-k+1)}\Big)^k \\ &\geq 2^{-k}\Big(1 - \frac{k}{N}\Big)^k \geq 2^{-k}\Big(1 - \frac{k^2}{N}\Big) \geq 2^{-k-1}.\end{aligned}$$

$\square$

3. THE MAIN ALGORITHM

In this section we prove our main approximate counting result, Theorem 1. We will make use of an algorithm with a weaker approximation guarantee, whose properties are stated in Lemma 18; we will prove this lemma in Section 4.

3.1. Sketch proof.

We first sketch a toy argument for the purpose of illustration. Suppose for convenience that our input hypergraph G has 2^ℓ vertices for some integer ℓ. Let t be a suitably large integer, and take independent uniformly random subsets $X_1, \dots, X_t \subseteq V(G)$ subject to $|X_i| = 2^{\ell-1}$ for all $i \in [t]$. It is not hard to show using Lemma 14 that $\mathbb{E}(e(G[X_i])) \approx e(G)/2^k$ for all i. Thus, using Hoeffding's inequality (Lemma 11), we can show that the total number of edges $\sum_{i=1}^t e(G[X_i])$ is concentrated around its mean of roughly $te(G)/2^k$. It follows that, with high probability, $(2^k/t)\sum_{i=1}^t e(G[X_i]) \approx e(G)$.

Repeating this expansion procedure yields the following (bad) algorithm. We maintain a list L of pairs (w, X), where $w \in \mathbb{Q}$ is positive and $X \subseteq V(G)$, and we preserve the invariant $\sum_{(w,X)\in L} we(G[X]) \approx e(G)$ with high probability. (We expect the quality of approximation to degrade as the algorithm runs, but we ignore this subtlety in our sketch.) Initially, we take $L = (1, V(G))$, which clearly satisfies this invariant. At each stage, for each pair $(w, X) \in L$, we independently choose t uniformly random subsets $X_1, \dots, X_t \subseteq X$ subject to $|X_i| = |X|/2$ for all i, as above. We then delete (w, X) from L and replace it by $(2^k w/t, X_1), \dots, (2^k w/t, X_t)$. Thus, as we proceed, L grows, but the sets X in L's entries become smaller, and the invariant $\sum_{(w,X)\in L} we(G[X]) \approx e(G)$ is maintained. Eventually, the entries of L become so small that for all $(w, X) \in L$, we can use cIND_G to count $e(G[X])$ quickly by brute force, and at this point we are done.

The problem with the algorithm described above is that in order to maintain the invariant with high probability, we must take $t = \Omega(\varepsilon^{-2}\log n)$, and to bring the vertex sets in L down to a manageable size we require $\Omega(\log n)$ expansion operations. Thus our final list will have length $(\varepsilon^{-2}\log n)^{\Omega(\log n)}$, resulting in an algorithm with superpolynomial running time. We avoid this problem by exploiting a statistical technique called importance sampling, previously applied to the $k = 2$ case by Beame et al. [6]. Given a coarse estimate of each $e(G[X_i])$, which need only be accurate to within a large multiplicative factor, this technique allows us to prune L to a manageable length in $\mathcal{O}(|L|)$ time, while maintaining the invariant $\sum_{(w,X)\in L} we(G[X]) \approx e(G)$ with high probability. Our algorithm for this, `Trim`, gives a substantially shorter list than the algorithm used in [6], thereby improving our running time.

To use this technique, we need the ability to find such coarse estimates. Beame et al. [6] gave a method to find these in the $k = 2$ case, which we substantially generalise to apply in our setting. Details of our coarse approximation algorithm, `Coarse`, can be found in Section 4.

Unlike [6], we also use these coarse estimates to improve the efficiency of our expansion procedure. The algorithm described above treats all pairs $(w, X) \in L$ equally, expanding each one into t smaller pairs. Thus L grows by a factor of t in a single expansion step. Our real algorithm will work differently. For each pair (w_i, X_i), we will choose the number t_i of replacement pairs according to our coarse estimate of $w_i e(G[X_i])$. We will take t_i to be large if (w_i, X_i) accounts for a large proportion of $\sum_{(w,X)\in L} we(G[X])$, and small otherwise; thus we only spend a lot of time processing a pair if it is "important" (see `Halve` in Section 3). This optimisation, together with the improved importance sampling procedure discussed above, drops our running time by a factor of roughly ε^{-2}. We therefore improve the results of [6] even in the $k = 2$ case.

3.2. The main algorithm.

We first prove a technical lemma, which should be read as follows. We are given the ability to sample from bounded probability distributions $\mathcal{D}_1, \dots, \mathcal{D}_q$ on $[0, \infty)$. We wish to estimate the sum of their means using as few samples as possible, and we are given access to a crude estimate of the mean of each $\mathcal{D}_i$ with multiplicative error b (for "bias"). Lemma 15 says that we can do so to within relative error ξ, with failure probability at most δ, by sampling t_i times from $\mathcal{D}_i$ for each $i \in [q]$. We will use this lemma in both `Trim` and `Halve`.

Lemma 15. *Let $0 < \xi, \delta < 1$, let $b \geq 1$, and let $M_1, \ldots, M_q > 0$. For all $i \in [q]$, let $\mathcal{D}_i$ be a probability distribution on $[0, M_i]$ with mean μ_i. For all $i \in [q]$, let $\hat{\mu}_i$ satisfy $0 < \hat{\mu}_i \leq \mu_i b$, and let*

$$t_i = \left\lceil \frac{4bM_i \log(2/\delta)}{\xi^2 \sum_j \hat{\mu}_j} \right\rceil.$$

Let $\{X_{i,j} \colon i \in [q], j \in [t_i]\}$ be independent random variables with $X_{i,j} \sim \mathcal{D}_i$. Then with probability at least $1 - \delta$,

$$\sum_{i=1}^{q} \sum_{j=1}^{t_i} \frac{X_{i,j}}{t_i} \in (1 \pm \xi) \sum_{i=1}^{q} \mu_i.$$

Note that while Lemma 15 does not require a lower bound on $\hat{\mu}_1, \ldots, \hat{\mu}_q$, without one it is useless as $\sum_i t_i$ may be arbitrarily large. When we apply Lemma 15, we will do so with $\mu_i/b \leq \hat{\mu}_i \leq \mu_i b$ for all $i \in [q]$.

Proof. We will apply a form of Bernstein's inequality (Lemma 12). Let

$$X = \sum_{i=1}^{q} \sum_{j=1}^{t_i} \frac{X_{i,j}}{t_i}, \qquad x = \sum_{i=1}^{q} \mu_i.$$

Thus we seek to prove $\mathbb{P}(X \in (1 \pm \xi)x) \geq 1 - \delta$. Note that $\mathbb{E}(X) = x$, and that

$$\sum_{i=1}^{q} \sum_{j=1}^{t_i} \mathbb{E}\big((X_{i,j}/t_i)^2\big) \leq \sum_{i=1}^{q} \sum_{j=1}^{t_i} \frac{1}{t_i^2} \mathbb{E}(M_i X_{i,j}) = \sum_{i=1}^{q} \frac{M_i \mu_i}{t_i}.$$

Let $M = \max\{M_i/t_i \colon i \in [q]\}$, so that $X_{i,j}/t_i \leq M$ for all i, j. Then by Lemma 12, applied to the variables X and $X_{i,j}/t_i$ with $z = \xi x$, it follows that

$$\begin{aligned} \mathbb{P}(|X - x| \geq \xi x) &\leq 2\exp\left(-\frac{3\xi^2 x^2}{6\sum_i \frac{M_i\mu_i}{t_i} + 2M\xi x}\right) \leq 2\exp\left(-\frac{3\xi^2 x^2}{2\max\left\{6\sum_i \frac{M_i\mu_i}{t_i},\ 2M\xi x\right\}}\right) \\ &= \max\left\{2\exp\left(-\frac{\xi^2 x^2}{4\sum_i \frac{M_i\mu_i}{t_i}}\right),\ 2\exp\left(-\frac{3\xi x}{4M}\right)\right\}. \end{aligned} \tag{1}$$

We now bound the exponents of each term in the max. By our choice of t_i's, we have

$$\frac{\xi^2 x^2}{4\sum_i \frac{M_i\mu_i}{t_i}} \geq \xi^2 x^2 \Big/ \left(4\sum_i M_i\mu_i \cdot \frac{\xi^2 \sum_j \hat{\mu}_j}{4bM_i \log(2/\delta)}\right) = \frac{bx^2 \log(2/\delta)}{\sum_i \mu_i \cdot \sum_j \hat{\mu}_j} = \frac{bx \log(2/\delta)}{\sum_j \hat{\mu}_j}.$$

Since $\hat{\mu}_i \leq \mu_i b$ for all i, we have $x \geq \sum_j \hat{\mu}_j / b$, so

$$\frac{\xi^2 x^2}{4\sum_i \frac{M_i\mu_i}{t_i}} \geq \log(2/\delta) \geq \ln(2/\delta). \tag{2}$$

Moreover, again by our choice of t_i's we have

$$M = \max\left\{\frac{M_i}{t_i} \colon i \in [q]\right\} \leq \max\left\{\frac{\xi^2 \sum_j \hat{\mu}_j}{4b\log(2/\delta)} \colon i \in [q]\right\} \leq \frac{\xi^2 x}{4\log(2/\delta)},$$

so

$$\frac{3\xi x}{4M} \geq \frac{3\log(2/\delta)}{\xi} > \ln(2/\delta). \tag{3}$$

The result therefore follows from (1), (2) and (3). □

Recall from our sketch proof in Section 3.1 that our algorithm will maintain a weighted list L of induced subgraphs of steadily decreasing size. For convenience, we will also include coarse estimates of the edge count of each graph in L. Rather than set out the format of this list each time we use it, we define it formally now.

Definition 16. *Let G be a hypergraph, let $i > 0$ be an integer, and let $b \geq 1$ be rational. Then a (G, b, y)-list is a list of triples $(w, S, \hat{e})$ such that w and $\hat{e}$ are positive rational numbers, $S \subseteq V(G)$ with $|S| = 2^y$, and $\hat{e}/b \leq e(G[S]) \leq \hat{e}b$. For any (G, b, y)-list L, we define*

$$Z(L) := \sum_{(w,S,\hat{e})\in L} we(G[S]).$$

Initially, we will take $L = ((1, V(G), \hat{e}))$ where $\hat{e}/b \leq e(G) \leq \hat{e}b$, so that $Z(L) = e(G)$. As the algorithm progresses, $Z(L)$ will remain a good approximation to $e(G)$, and eventually we will be able to compute it efficiently. We are now ready to set out our importance sampling algorithm, `Trim`, which we will use to keep the length of L low.

Algorithm $\texttt{Trim}(G, b, y, L, \xi, \delta)$**.**

Input: G is an n-vertex k-hypergraph, where n is a power of 2, to which `Trim` has (only) coloured oracle access. b is a rational number with $b \geq 1$, and y is a positive integer. L is a (G, b, y)-list with $1/2 \leq Z(L) \leq 2n^k$ and $|L| \leq n^{11k}$. δ is a rational number with $0 < \delta < 1$, and ξ is a rational number with $n^{-2k} \leq \xi < 1$.

Behaviour: $\texttt{Trim}(G, b, y, L, \xi, \delta)$ outputs a (G, b, y)-list L' satisfying the following properties.

(a) $|L'| \leq 33k \log(4nb) + 32b^2 \log(2/\delta)/\xi^2$.

(b) With probability at least $1 - \delta$, $Z(L') \in (1 \pm \xi)Z(L)$.

(T1) Calculate $a \leftarrow \lfloor 15k \log(4nb) \rfloor + 1$ and

$$L_i \leftarrow \{(w, S, \hat{e}) \in L \colon 2^{i-1} \leq w\hat{e} < 2^i\} \text{ for each } -a \leq i \leq a.$$

(Every significant entry of L will be contained in exactly one L_i, and entries $(w, S, \hat{e}) \in L_i$ satisfy $w\hat{e} \approx 2^i$.)

(T2) For each $-a \leq i \leq a$, calculate

$$t_i \leftarrow \left\lceil \frac{16b^2 2^i |L_i| \log(2/\delta)}{\xi^2 W} \right\rceil, \text{ where } W := \sum_{(w,S,\hat{e})\in L} w\hat{e}.$$

(T3) For each $-a \leq i \leq a$, calculate a multiset L'_i as follows. If $|L_i| \leq t_i$, let $L'_i \leftarrow L_i$. Otherwise, sample t_i entries $(w_{i,1}, S_{i,1}, \hat{e}_{i,1}), \ldots, (w_{i,t_i}, S_{i,t_i}, \hat{e}_{i,t_i})$ from L_i independently and uniformly at random, let $w'_{i,j} \leftarrow w_{i,j}|L_i|/t_i$, and let $L'_i \leftarrow \{(w'_{i,j}, S_{i,j}, \hat{e}_{i,j}) \colon j \in [t_i]\}$.

(T4) Form L' by concatenating the multisets $\{L'_i \colon -a \leq i \leq a\}$ in arbitrary order, and return L'.

Note that `Trim` improves significantly on the importance sampling algorithm of [6, Lemma 2.5], which in this setting outputs a list of length $\Omega(kb^4 \log(1/\delta) \log(nb)/\xi^2)$.

Lemma 17. $\texttt{Trim}(G, b, y, L, \xi, \delta)$ *behaves as claimed above, has running time $\mathcal{O}(|L|k \log(nb/\delta))$, and does not invoke* cIND_G.

Proof. Running time. It is clear that $\texttt{Trim}(G, b, y, L, \xi, \delta)$ does not invoke cIND_G. Recall that we work with the word-RAM model, so we can carry out elementary arithmetic operations on $\mathcal{O}(\log n)$-sized numbers in $\mathcal{O}(1)$ time. Thus step (T1) takes time $\mathcal{O}(a|L|)$, and step (T2) takes time $\mathcal{O}(a \log(1/\delta) + a|L|)$. Since $|L'_i| = \min\{|L_i|, t_i\}$, steps (T3) and (T4) take time $\mathcal{O}(a|L|)$. The required bounds follow.

Correctness. Since every entry of L' is an entry of L (perhaps with a different first element), and L is a (G, b, y)-list, L' is also a (G, b, y)-list. We next prove (a). We have

$$\begin{aligned} |L'| &= \sum_{|i|\le a} |L'_i| \le \sum_{|i|\le a} t_i \le \sum_{|i|\le a} \Big(1 + \frac{16b^2 2^i |L_i| \log(2/\delta)}{\xi^2 W}\Big) \\ &= 2a + 1 + \frac{16b^2 \log(2/\delta)}{\xi^2 W} \sum_{|i|\le a} 2^i |L_i|. \end{aligned} \tag{4}$$

Recall from the definition of L_i that, for all $(w, S, \hat{e}) \in L_i$, we have $w\hat{e} \ge 2^{i-1}$, so

$$\sum_{|i|\le a} 2^i |L_i| \le \sum_{|i|\le a} \Big(2 \sum_{(w,S,\hat{e})\in L_i} w\hat{e}\Big) = 2W.$$

It therefore follows from (4) that $|L'| \le 2a + 1 + 32b^2 \log(2/\delta)/\xi^2$, so property (a) holds.

It remains to prove property (b). This will follow easily from Lemma 15. Before we can apply it, however, we must set our notation and prove the conditions of the lemma hold. Let

$$\mathcal{I} := \{-a \le i \le a \colon |L_i| > t_i\}$$

be the set of indices i such that we choose L'_i by sampling elements $(w_{i,j}, S_{i,j}, \hat{e}_{i,j})$ of L_i. For each $i \in \mathcal{I}$ and $j \in [t_i]$, let

$$\begin{aligned} X_{i,j} &:= w_{i,j} e(G[S_{i,j}]) |L_i|, & M_i &:= 2^i b |L_i|, \\ \mu_i &:= \sum_{(w,S,\hat{e})\in L_i} w e(G[S]), & \hat{\mu}_i &:= \sum_{(w,S,\hat{e})\in L_i} w\hat{e}. \end{aligned}$$

For all i and j, it is clear that $X_{i,j} \ge 0$. Moreover, since L is a (G, b, y)-list and $(w_{i,j}, S_{i,j}, \hat{e}_{i,j}) \in L$, we have $e(G[S_{i,j}]) \le b\hat{e}_{i,j}$; thus by the definitions of L_i and $X_{i,j}$ we have

$$X_{i,j} \le b w_{i,j} \hat{e}_{i,j} |L_i| \le 2^i b |L_i| = M_i.$$

It is also true that $\mathbb{E}(X_{i,j}) = \mu_i$, that $0 \le \hat{\mu}_i \le \mu_i b$, that the $X_{i,j}$'s are independent, and that

$$\Big\lceil \frac{4bM_i \log(2/\delta)}{(\xi/2)^2 \sum_\ell \hat{\mu}_\ell} \Big\rceil = \Big\lceil \frac{16b^2 2^i |L_i| \log(2/\delta)}{\xi^2 W} \Big\rceil = t_i.$$

It therefore follows from Lemma 15 that with probability at least $1 - \delta$,

$$\sum_{i\in\mathcal{I}} \sum_{j=1}^{t_i} \frac{X_{i,j}}{t_i} \in (1 \pm \xi/2) \sum_{i\in\mathcal{I}} \mu_i. \tag{5}$$

Suppose this event occurs; then we will show that $Z(L') \in (1 \pm \xi) Z(L)$, as in (c).

Plugging our definitions into (5), we see that

$$\sum_{i\in\mathcal{I}} \sum_{j=1}^{t_i} \frac{X_{i,j}}{t_i} = \sum_{i\in\mathcal{I}} \sum_{j=1}^{t_i} \frac{w_{i,j}|L_i|}{t_i} e(G[S_{i,j}]) = \sum_{i\in\mathcal{I}} \sum_{(w,S,\hat{e})\in L'_i} w e(G[S]),$$

and

$$\sum_{i\in\mathcal{I}} \mu_i = \sum_{i\in\mathcal{I}} \sum_{(w,S,\hat{e})\in L_i} w e(G[S]),$$

so

$$\sum_{i\in\mathcal{I}} \sum_{(w,S,\hat{e})\in L'_i} w e(G[S]) \in (1 \pm \xi/2) \sum_{i\in\mathcal{I}} \sum_{(w,S,\hat{e})\in L_i} w e(G[S]).$$

We have $L'_i = L_i$ for all $i \in \{-a, \dots, a\} \setminus \mathcal{I}$, so it follows that

$$\sum_{|i|\le a} \sum_{(w,S,\hat{e})\in L'_i} we(G[S]) \in (1 \pm \xi/2) \sum_{|i|\le a} \sum_{(w,S,\hat{e})\in L_i} we(G[S]). \tag{6}$$

For all $(w, S, \hat{e}) \in L$, since L is a (G, b, y)-list with $Z(L) \le 2n^k$, we have

$$w\hat{e} \le bwe(S) \le bZ(L) \le 2bn^k < 2^a.$$

Thus for all $(w, S, \hat{e}) \in L \setminus \bigcup_{|i|\le a} L_i$, we have $w\hat{e} \le 2^{-a}$. It follows from (6) that

$$Z(L') = \sum_{|i|\le a} \sum_{(w,S,\hat{e})\in L'_i} we(G[S]) \in (1 \pm \xi/2)Z(L) \pm 2^{-a}|L|.$$

Observe that by hypothesis, $\xi Z(L)/2 \ge n^{-2k}/4$ and $2^{-a}|L| \le n^{-2k}/4$. Hence $Z(L') \in (1 \pm \xi)Z(L)$, as required. □

We next state the behaviour of our coarse approximate counting algorithm; we will prove the following lemma in Section 4.

Lemma 18. *There is a randomised algorithm* $\texttt{Coarse}(G, \delta)$ *with the following behaviour. Suppose G is an n-vertex k-hypergraph to which* `Coarse` *has (only) coloured oracle access, where n is a power of two, and suppose $0 < \delta < 1$. Then in time $\mathcal{O}(\log(1/\delta)k^{3k} n \log^{2k+2} n)$, and using at most $\mathcal{O}(\log(1/\delta)k^{3k} \log^{2k+2} n)$ queries to* cIND_G, $\texttt{Coarse}(G, \delta)$ *outputs a rational number $\hat{e}$. Moreover, with probability at least $1 - \delta$,*

$$\frac{e(G)}{2(4k \log n)^k} \le \hat{e} \le e(G) \cdot 2(4k \log n)^k.$$

Using `Coarse`, we now set out our algorithm for decreasing the size of elements in our (G, b, y)-list L, that is, turning it into a $(G, b, y - 1)$-list L' with $Z(L') \approx Z(L)$.

Algorithm $\texttt{Halve}(G, b, y, L, \xi, \delta)$**.**

Input: G is an n-vertex k-hypergraph, where n is a power of 2, to which `Halve` has (only) coloured oracle access. b is a rational number with $b \ge 2(4k \log n)^k$, and y is a positive integer with $2^{y-1} \ge 2k^2$. L is a (G, b, y)-list. ξ and δ are rational numbers with $0 < \xi,\ \delta < 1$.

Behaviour: $\texttt{Halve}(G, b, y, L, \xi, \delta)$ outputs a list L', which satisfies the following properties with probability at least $1 - \delta$.

(a) L' is a $(G, b, y - 1)$-list.
(b) $|L'| \le |L| + 2^{k+3}b^2 \log(4/\delta)/\xi^2$.
(c) $Z(L') \in (1 \pm \xi)Z(L)$.

(H1) Write $L =: \{(w_i, S_i, \hat{e}_i) \colon 1 \le i \le |L|\}$. Calculate

$$p \leftarrow \binom{2^y - k}{2^{y-1} - k} \Big/ \binom{2^y}{2^{y-1}}, \qquad W \leftarrow \sum_{i=1}^{|L|} w_i \hat{e}_i,$$

$$\text{and } t_i \leftarrow \left\lceil \frac{4b^2 w_i \hat{e}_i \log(4/\delta)}{p\xi^2 W} \right\rceil \text{ for all } 1 \le i \le |L|.$$

(H2) For all $1 \le i \le |L|$, sample subsets $S_{i,1}, \dots, S_{i,t_i} \subseteq S_i$ independently and uniformly at random subject to $|S_{i,j}| = 2^{y-1}$. Then calculate $w'_i \leftarrow w_i/pt_i$ and

$$L'_i \leftarrow \Big\{ \Big(w'_i, S_{i,j}, \texttt{Coarse}\big(G[S_{i,j}], \delta/2\textstyle\sum_i t_i\big) \Big) \colon 1 \le i \le |L|,\ j \in [t_i] \Big\}.$$

(H3) Form L' by concatenating the multisets $\{L'_i \colon 1 \le i \le |L|\}$ in arbitrary order and removing any entries $(w, S, \hat{e})$ with $\hat{e} = 0$, and return L'.

Lemma 19. $\texttt{Halve}(G, b, y, L, \xi, \delta)$ *behaves as claimed above. Moreover, writing*

$$\lambda = |L| + \frac{2^k b^2 \log(1/\delta)}{\xi^2}, \qquad T = \lambda \log(\lambda/\delta) k^{3k} \log^{2k+2} n,$$

$\texttt{Halve}(G, b, y, L, \xi, \delta)$ *has running time* $\mathcal{O}(nT)$ *and invokes* cIND_G *at most* $\mathcal{O}(T)$ *times.*

Proof. Running time. The running time and oracle usage are both dominated by the invocations of `Coarse` in step (H2). We first bound the number $\sum_i t_i$ of such invocations. We have

$$\sum_{i=1}^{|L|} t_i \le |L| + \sum_{i=1}^{|L|} \frac{4b^2 w_i \hat{e}_i \log(4/\delta)}{p\xi^2 W} = |L| + \frac{4b^2 \log(4/\delta)}{p\xi^2}. \tag{7}$$

Since $2^{y-1} \ge 2k^2$, by a standard binomial coefficient bound (Lemma 14), we have $p \ge 2^{-k-1}$. Thus (7) implies

$$\sum_{i=1}^{|L|} t_i \le |L| + \frac{2^{k+3} b^2 \log(4/\delta)}{\xi^2} = \Theta(\lambda). \tag{8}$$

By Lemma 18, writing $T' = \log(\lambda/\delta) k^{3k} \log^{2k+2} 2^{y-1}$, `Coarse` has running time $\mathcal{O}(2^{y-1}T')$ and invokes cIND_G $\mathcal{O}(T')$ times. Since L is a (G, b, y)-list, we have $2^{y-1} \le n$ and so the claimed bounds follow from (8).

Correctness. Let $\mathcal{E}_1$ be the event that $Z(L') \in (1 \pm \xi) Z(L)$, and let $\mathcal{E}_2$ be the event that every invocation of `Coarse` in step (H2) succeeds. We will show that $\mathbb{P}(\mathcal{E}_1 \cap \mathcal{E}_2) \ge 1 - \delta$, and that properties (a)–(c) hold whenever $\mathcal{E}_1 \cap \mathcal{E}_2$ occurs.

Bounding $\mathbb{P}(\mathcal{E}_1 \cap \mathcal{E}_2)$: To bound $\mathbb{P}(\mathcal{E}_1)$ below, we will apply Lemma 15. We first set up our notation and show that the relevant assumptions hold. For all $1 \le i \le |L|$ and all $j \in [t_i]$, let

$$X_{i,j} := w_i e(G[S_{i,j}])/p, \qquad M_i := w_i \hat{e}_i b/p,$$
$$\mu_i := w_i e(G[S_i]), \qquad \hat{\mu}_i := w_i \hat{e}_i.$$

For all i and j, it is clear that $X_{i,j} \ge 0$. Moreover, since $(w_i, S_i, \hat{e}_i) \in L$, we have $e(G[S_{i,j}]) \le e(G[S_i]) \le \hat{e}_i b$, so $X_{i,j} \le M_i$. Since L is a (G, b, y)-list, we have $|S_i| = 2^y$, so p is the probability that any given edge in $G[S_i]$ survives in $G[S_{i,j}]$; thus $\mu_i = \mathbb{E}(X_{i,j})$. The $X_{i,j}$'s are independent, we have $0 \le \hat{\mu}_i \le \mu_i b$, and we have

$$\left\lceil \frac{4bM_i \log(4/\delta)}{\xi^2 \sum_\ell \hat{\mu}_\ell} \right\rceil = \left\lceil \frac{4b^2 w_i \hat{e}_i \log(4/\delta)}{p\xi^2 W} \right\rceil = t_i.$$

It therefore follows from Lemma 15 that with probability at least $1 - \delta/2$,

$$\sum_{i=1}^{|L|} \sum_{j=1}^{t_i} \frac{X_{i,j}}{t_i} \in (1 \pm \xi) \sum_{i=1}^{|L|} \mu_i. \tag{9}$$

Plugging our definitions in, we see (9) implies that $Z(L') \in (1 \pm \xi) Z(L)$. Thus

$$\mathbb{P}(\mathcal{E}_1) \ge 1 - \delta/2. \tag{10}$$

By the correctness of `Coarse` (Lemma 18) and a union bound over all $1 \le i \le |L|$ and all $j \in [t_i]$, we have $\mathbb{P}(\mathcal{E}_2) \ge 1 - \delta/2$. By a union bound with (10), we therefore have $\mathbb{P}(\mathcal{E}_1 \cap \mathcal{E}_2) \ge 1 - \delta$ as claimed.

Properties (a)–(c) hold: Suppose $\mathcal{E}_1 \cap \mathcal{E}_2$ occurs. For every entry $(w, S, \hat{e})$ of L', w and $\hat{e}$ are positive rational numbers and $S \subseteq V(G)$ with $|S| = 2^{y-1}$. Since $\mathcal{E}_2$ occurs and $b \ge 2(4k \log n)^k$, by the correctness

of `Coarse` (Lemma 18) we have $\hat{e}/b \le e(G[S]) \le \hat{e}b$. Thus L is a $(G, b, y-1)$-list as required by property (a). We have $|L'| = \sum_i t_i$, so (8) implies that property (b) holds. Finally, since $\mathcal{E}_1$ occurs, property (c) holds. Thus properties (a)–(c) all hold whenever $\mathcal{E}_1 \cap \mathcal{E}_2$ occurs, which we have already shown happens with probability at least $1-\delta$. □

We now state our main algorithm.

Algorithm `HelperCount`(G, ε)**.**

Input: G is an n-vertex k-hypergraph, where n is a power of 2. `HelperCount` has (only) coloured oracle access to G, and ε is a rational number with $0 < \varepsilon < 1/2$.

Behaviour: `HelperCount`(G, ε) outputs a rational number $\hat{e}$ such that with probability at least 2/3, $\hat{e} \in (1 \pm \varepsilon)e(G)$.

(A1) If $\varepsilon < n^{-k}$, or if $n \le 500$, then return $\sum_{Y \subseteq V(G),\ |Y|=k}(1 - \text{cIND}_G(Y))$.

(A2) If `Coarse`$(G, \delta) = 0$, return 0. Otherwise, let

$$I \leftarrow \log n - \lceil \log(2k^2) \rceil, \qquad b \leftarrow 2(4k \log n)^k,$$
$$\xi \leftarrow \varepsilon/4I, \qquad \delta \leftarrow 1/3(2I+1),$$
$$L \leftarrow \big\{\big(1, V(G), \texttt{Coarse}(G, \delta)\big)\big\}.$$

(A3) For $i = 1$ to I:

(We will maintain the invariant that L is a $(G, b, \log n - (i-1))$-list with $Z(L) \in (1 \pm \xi)^{2i} e(G)$, and that $|L|$ is suitably small (see proof of Lemma 20). Note that this is trivially satisfied at the start of the loop.)

(A4) Update $L \leftarrow$ `Halve`$(G, b, \log n - (i-1), L, \xi, \delta)$.
(This step turns L into a $(G, b, \log n - i)$-list.)

(A5) Update $L \leftarrow$ `Trim`$(G, b, \log n - i, L, \xi, \delta)$.
(This step reduces the length of L.)

(A6) For each entry $(w, S, \hat{e}) \in L$, calculate

$$e_S \leftarrow \sum_{\substack{Y \subseteq S \\ |Y|=k}} (1 - \text{cIND}_G(Y)).$$

(A7) Output $\sum_{(w,S,\hat{e}) \in L} w e_S$.

Lemma 20. *With probability at least $2/3$, `HelperCount`(G, ε) outputs a rational number $\hat{e} \in (1 \pm \varepsilon)e(G)$ as claimed above, has running time $\mathcal{O}(\varepsilon^{-2} k^{6k} n \log^{4k+7} n)$, and invokes cIND_G at most $\mathcal{O}(\varepsilon^{-2} k^{6k} \log^{4k+7} n)$ times.*

Proof. Correctness. Lemma 18 implies that correctness holds if `HelperCount`(G, ε) outputs on step (A1) or (A2), so suppose that it does not. For all integers $i \ge 0$, let $\pi(i)$ be the statement that L satisfies the following properties.

(i) L is a $(G, b, \log n - i)$-list.
(ii) $Z(L) \in (1 \pm \xi)^{2i} e(G)$.
(iii) $|L| \le 33k \log(4nb) + 32b^2 \log(2/\delta)/\xi^2$.

We will prove that with probability at least $2/3$, $\pi(i)$ holds at the end of the ith iteration of loop (A3) for all $i \in [I]$. Suppose this is true: we will show that correctness follows. With probability at least $2/3$, $\pi(I)$ holds when we exit the loop. In this case, the final value of L satisfies $Z(L) \in (1 \pm \xi)^{2I} e(G)$ by (ii). We have $(1-\xi)^{2I} \ge 1 - 2I\xi$, and $(1+\xi)^{2I} \le e^{2I\xi} \le 1 + 4I\xi$ (since $4I\xi = \varepsilon < 1$), so

$$Z(L) \in (1 \pm 4I\xi)e(G) = (1 \pm \varepsilon)e(G).$$

Moreover, in step (A6) we have $e_S = e(G[S])$ for all $(w, S, \hat{e}) \in L$, so $Z(L)$ is the output.

It remains to prove that with probability at least $2/3$, $\pi(i)$ holds at the end of the ith iteration of loop (A3) for all $i \in [I]$. Let $\mathcal{E}_0$ be the event that `Coarse` behaves correctly in step (A2); note that $\mathbb{P}(\mathcal{E}_0) \geq 1 - \delta$ by the correctness of `Coarse` (Lemma 18). For all $i \in [I]$, let $\mathcal{E}_i$ be the event that `Halve` behaves correctly in the ith iteration of step (A4) and `Trim` behaves correctly in the ith iteration of step (A5). (If the input restrictions of `Halve` or `Trim` are violated on the ith iteration, then $\mathcal{E}_i$ occurs automatically.) By correctness of `Halve` and `Trim` (Lemmas 19 and 17), we have $\mathbb{P}(\mathcal{E}_i \mid \mathcal{E}_0, \ldots, \mathcal{E}_{i-1}) \geq 1 - 2\delta$. Thus by a union bound over all $0 \leq i \leq I$, we have

$$\mathbb{P}\Big(\bigcap_{i=0}^{I} \mathcal{E}_i\Big) \geq 1 - (2I+1)\delta = 2/3.$$

It therefore suffices to show that when $\bigcap_j \mathcal{E}_j$ occurs, $\pi(i)$ holds at the end of the ith iteration of loop (A3) for all $i \in [I]$.

At the start of the first iteration of loop (A3), when $i = 1$, L is a $(G, b, \log n)$-list since $\mathcal{E}_0$ occurs, $Z(L) = e(G)$, and $|L| = 1$. Thus $\pi(0)$ holds. Let $i \in [I]$, and suppose that $\pi(i-1)$ holds at the start of the ith iteration of loop (A3). Let L_i be the value of L at the start of the ith iteration, let L_i' be the value of L after executing step (A4), and let L_{i+1} be the value of L after executing step (A5).

By property (i), L_i is a $(G, b, \log n - (i-1))$-list, where by our choice of I we have $2^{\log n - i} \geq 2k^2$. Since $\mathcal{E}_i$ occurs, it follows by the correctness of `Halve` (Lemma 19) that L_i' is a $(G, b, \log n - i)$-list with

$$\begin{aligned} Z(L_i') &\in (1 \pm \xi) Z(L_i), \\ |L_i'| &\leq |L_i| + 2^{k+3} b^2 \log(4/\delta)/\xi^2. \end{aligned}$$

We next show that $1/2 \leq Z(L_i') \leq 2n^k$, that $|L_i'| \leq n^{11k}$, and that $\xi \geq n^{-2k}$, as required by `Trim`. Since property (ii) holds for $Z(L_i)$, we have $Z(L_i') \in (1 \pm \xi)^{2i+1} e(G) \subseteq (1 \pm \varepsilon) e(G)$. Since `HelperCount`$(G, \varepsilon)$ did not halt at (A2), we have $1 \leq e(G) \leq n^k$; since $\varepsilon < 1/2$, it follows that $1/2 \leq Z(L_i') \leq 2n^k$. Since `HelperCount`$(G, \varepsilon)$ did not halt at (A1), we have $n \geq 500$ and hence $n \geq 50 \log n$. We also have $\varepsilon \geq n^{-k}$ and $k \leq n$. Hence:

$$\begin{aligned} &b^2 \leq n^{4k}; \qquad 2^k \leq n^k; \qquad \log(4/\delta) \leq \log(24 \log n) \leq n; \\ &\log(4nb) \leq 6k \log n \leq n^2; \qquad 1/\xi^2 \leq 16 (\log n)^2 n^{2k} \leq n^{4k}. \end{aligned}$$

Since property (iii) holds for L_i, it follows that

$$|L_i'| \leq 33k \log(4nb) + 40 \cdot 2^k b^2 \log(4/\delta)/\xi^2 \leq 33n^3 + 40n^{9k+1} \leq n^{10k}.$$

We have therefore shown that L_i' and ξ satisfy the input restrictions of `Trim`. Since $\mathcal{E}_i$ occurs, by the correctness of `Trim` (Lemma 17) it follows that L_{i+1} is a $(G, b, \log n - i)$-list with

$$\begin{aligned} Z(L_{i+1}) &\in (1 \pm \xi) Z(L_i') \subseteq (1 \pm \xi)^2 Z(L_i) \subseteq (1 \pm \xi)^{2(i+1)} e(G), \\ |L_{i+1}| &\leq 33k \log(4nb) + 32b^2 \log(2/\delta)/\xi^2. \end{aligned}$$

Thus properties (i)–(iii) hold for L_{i+1}, as required.

Running time and oracle queries. If step (A1) is executed, so that $\varepsilon < n^{-k}$ or $n \leq 500$, then the algorithm runs in time $\mathcal{O}(n^k) = \mathcal{O}(\varepsilon^{-1})$ and uses $\mathcal{O}(n^k) = \mathcal{O}(\varepsilon^{-1})$ oracle queries, so our claimed bounds hold. Suppose instead step (A1) is not executed, so that $\varepsilon \geq n^{-k}$. Recall that $\bigcap_i \mathcal{E}_i$ holds with probability at least $2/3$. Suppose this occurs. The bottleneck in both running time and oracle invocations is then step (A4). For legibility, we give the time and oracle requirements of the other steps in the following table, giving justifications in the paragraph below. We write $\Lambda = 33k \log(4nb) + 32b^2 \log(2/\delta)/\xi^2$ for the upper bound on $|L|$ in property (iii) of our invariant π.

Step number	Running time	Oracle calls
(A1)	$\mathcal{O}(1)$	None
(A2)	$\mathcal{O}\big(k^{3k} n \log^{2k+2} n \log(1/\delta)\big)$	$\mathcal{O}\big(k^{3k} \log^{2k+2} n \log(1/\delta)\big)$
(A3)	$\mathcal{O}(I)$	None
(A5)	$\mathcal{O}\Big(I\big(\Lambda + 2^k b^2 \xi^{-2} \log(1/\delta)\big) k^2 \log n\Big)$	None
(A6)	$\mathcal{O}(\Lambda(4k^2)^k)$	$\mathcal{O}(\Lambda(4k^2)^k)$
(A7)	$\mathcal{O}(\Lambda)$	None

For step (A1), we use the fact that $\varepsilon \geq n^{-k}$ and so the conditional does not trigger. For step (A2), we use the fact that n is a power of 2 (so computing $\log n$ is easy) and the time bounds on `Coarse` (Lemma 18). For step (A5), we first observe that the step is executed I times. We then apply the time bounds on `Trim` (Lemma 17), together with property (iii) and the fact that `Halve` adds at most $2^{k+3} b^2 \xi^{-2} \log(4/\delta)$ to the length of L (see Lemma 19). (Note that $k \log(nb/\delta) = \mathcal{O}(k^2 \log n)$.) For step (A6), we use the fact that after the loop of (A3), L is a $(G, b, \log n - I)$-list, so each entry $(w, S, \hat{e})$ of L has $|S| = 2^{\log n - I} \leq 4k^2$.

We now consider step (A4), which is executed I times. From the time bounds of `Halve` (Lemma 19), it follows that the total running time of step (A4) is $\mathcal{O}(nT)$ and the total number of oracle accesses is $\mathcal{O}(T)$ times, where

$$T = \mathcal{O}\big(I\lambda \log(\lambda/\delta) k^{3k} \log^{2k+2} n\big), \qquad \lambda = \Lambda + 2^k b^2 \xi^{-2} \log(1/\delta).$$

This clearly dominates everything in the table. Observe that $\Lambda = \mathcal{O}(2^k b^2 \xi^{-2} \log(1/\delta))$, so $\lambda = \mathcal{O}(2^k b^2 \xi^{-2} \log(1/\delta))$ also. Since $\varepsilon \geq n^{-k}$,

$$\log(\lambda/\delta) = \mathcal{O}\Big(k + k \log(k \log n) + \log I + \log(1/\varepsilon) + \log(1/\delta)\Big) = \mathcal{O}(k \log n).$$

Thus

$$\begin{aligned} T &= \mathcal{O}\Big(I 2^k b^2 \xi^{-2} \log(1/\delta) k^{3k+1} \log^{2k+3} n\Big) \\ &= \mathcal{O}\Big(\log n \cdot 2^k \cdot (k \log n)^{2k} \cdot \varepsilon^{-2} \log^2 n \cdot \log\log n \cdot k^{3k+1} \log^{2k+3} n\Big) \\ &= \mathcal{O}\big(\varepsilon^{-2} k^{6k} \log^{4k+7} n\big), \end{aligned}$$

and the claimed bounds follow. □

We now prove Theorem 1.

Theorem 1 (restated)**.** *There is a randomised algorithm* $\texttt{Count}(G, \varepsilon, \delta)$ *with the following behaviour. Suppose G is an n-vertex k-hypergraph, and that* `Count` *has coloured oracle access to G. Suppose ε and δ are rational with $0 < \varepsilon, \delta < 1$. Then, writing $T = \log(1/\delta)\varepsilon^{-2} k^{6k} \log^{4k+7} n$: in time $\mathcal{O}(nT)$, and using at most $\mathcal{O}(T)$ queries to* cIND_G*,* $\texttt{Count}(G, \varepsilon, \delta)$ *outputs a rational number $\hat{e}$. With probability at least $1 - \delta$, we have $\hat{e} \in (1 \pm \varepsilon) e(G)$.*

Proof. To evaluate $\texttt{Count}(G, \varepsilon, \delta)$, we first make n into a power of two by adding at most n isolated vertices to G; note that this does not impede the evaluation of cIND_G. We then run $\texttt{HelperCount}(G, \min\{\varepsilon, 1/3\})$ a total of $36\lceil \ln(2/\delta) \rceil$ times and return the median result $\hat{e}$. If some invocation of $\texttt{HelperCount}(G, \min\{\varepsilon, 1/3\})$ takes more than $\Theta(\varepsilon^{-2} k^{6k} n \log^{4k+7} n)$ time, or invokes cIND_G more than $\Theta(\varepsilon^{-2} k^{6k} \log^{4k+7} n)$ times, we halt execution and consider the output to be -1.

It is immediate that this algorithm satisfies our stated time bounds. Moreover, $\hat{e} \in (1 \pm \varepsilon) e(G)$ unless at least half our invocations of `HelperCount` fail. The number of such failures is dominated above by a binomial variable N with mean $12\lceil \ln(2/\delta) \rceil$, so by a standard Chernoff bound (namely Lemma 13(i)) we have

$$\mathbb{P}\big(\hat{e} \notin (1 \pm \varepsilon) e(G)\big) \leq \mathbb{P}\big(N \geq 18\lceil \ln(2/\delta) \rceil\big) \leq \mathbb{P}\Big(\big|N - \mathbb{E}(N)\big| \geq \frac{1}{2}\mathbb{E}(N)\Big) \leq 2e^{-\lceil \ln(2/\delta) \rceil} \leq \delta,$$

as required. □

4. Coarse approximate counting

In this section, we prove Lemma 18. Throughout, we fix the input graph G to be an n-vertex k-hypergraph to which we have (only) coloured oracle access, where n is a power of two.

4.1. **Sketch proof.** The heart of our algorithm will be a subroutine to solve the following simpler "gap-version" of the problem. Given a k-partite k-hypergraph G and a guess $M \geq 0$, we ask: Does G have more than M edges? We wish to answer correctly with high probability provided that either G has at least M edges, or G has significantly fewer than M edges, namely at most γM edges with $\gamma = 1/(2^{3k+1}k^{2k}\log^k n)$. Suppose we can solve this problem probabilistically, perhaps outputting `Yes` with probability at least $1/50$ if $e(G) \geq M$ (which we call *completeness*) and outputting `Yes` with probability at most $1/100$ if $e(G) \leq \gamma M$ (which we call *soundness*). We then apply probability amplification to substantially reduce the failure probability, and use binary search to find the least M such that our output is `Yes`— with high probability, this will approximate $e(G)$ when our input k-hypergraph is k-partite. We then generalise our algorithm to arbitrary inputs using random colour-coding. These parts of the algorithm are fairly standard, so in this sketch proof we will only solve the gap-problem. (We implement this sketch below as the `VerifyGuess` algorithm.)

Let G be a k-partite k-hypergraph with vertex classes $X_1, \ldots, X_k$. The basic idea of the algorithm is to randomly remove vertices from G to form a new graph H in such a way that each edge survives with probability roughly $1/M$, and then query the coloured independence oracle and output `Yes` if and only if at least one edge remains. If G has at most γM edges, then a union bound implies we are likely to output `No` (soundness); if G has at least M edges, then in expectation at least one edge survives the removal, so we hope to output `Yes` (completeness). Unfortunately, the number of edges remaining in H need not be concentrated around its expectation — for example, if every edge of G is incident to a single vertex v — so we must be very careful if this hope is to be realised.

Suppose for the moment that $k = 2$, so that G is a bipartite graph with vertex classes X_1 and X_2. Then we will form $X_1' \subseteq X_1$ by including each vertex independently with probability p_1, and $X_2' \subseteq X_2$ by including each vertex independently with probability p_2. Each edge survives with probability p_1p_2, so we require $p_1p_2 \leq 1/M$ to ensure soundness. To ensure completeness, we would then like to choose p_1 and p_2 such that $G[X_1', X_2']$ is likely to contain an edge whenever $e(G) \geq M$.

To see that such a pair (p_1, p_2) exists, we first partition the vertices in X_1 according to their degree: For $1 \leq d \leq \log n$, let X_1^d be the set of vertices v with $2^{d-1} \leq d(v) < 2^d$. By the pigeonhole principle, there exists some D such that X_1^D is incident to at least $e(G)/\log n$ edges. Then we take $p_1 = 2^D/M$ and $p_2 = 1/2^D$. We certainly have $p_1p_2 \leq 1/M$. Suppose $e(G) \geq M$. Since X_1^D is incident to at least $e(G)/\log n$ edges, we have $|X_1^D| \geq M/2^D \log n$, so with reasonable probability X_1' contains a vertex $v_1 \in X_1^D$. Then v_1 has degree roughly 2^D in X_2, so again with reasonable probability X_2' contains a vertex adjacent to it.

There is one remaining obstacle: Since we only have coloured oracle access to G, we do not know what D is! Fortunately, since there are only $\mathcal{O}(\log n)$ possibilities, we can simply try them all in turn, and output `Yes` if any one of them yields a pair X_1', X_2' such that $G[X_1', X_2']$ contains an edge. (It is not hard to tune the parameters so that this doesn't affect soundness.) This is essentially the argument used by Beame et al. [6].

When we try to generalise this approach to k-hypergraphs, we hit a problem. For illustration, take $k = 3$ and suppose $e(G) \geq M$. Then we wish to guess a vector (p_1, p_2, p_3) such that $p_1p_2p_3 \leq 1/M$ and, with reasonable probability, $G[X_1', X_2', X_3']$ contains an edge. As in the $k = 2$ case, we can guess an integer $0 \leq D \leq 2\log n$ such that a large proportion of G's edges are incident to a vertex in X_1 of degree roughly 2^D. Also as in the $k = 2$ case, if we take $p_1 = 2^D/M$ then it is reasonably likely that X_1' will contain a vertex of degree roughly 2^D, say v_1. But we cannot iterate this process — the structure of $G[v_1, X_2, X_3]$, and hence the "correct" value of p_2, depends very heavily on v_1. So for example, when we test the two guesses $(2^D/M, 1/2^D, 1)$ and $(2^D/M, 1, 1/2^D)$, we wish to ensure that the value of v_1 is the same in each test. This is the reason for step (C1) in the following algorithm; it is important that we do not choose new random subsets of $X_1, \ldots, X_k$ independently with each iteration of step (C2).

Algorithm `VerifyGuess`$(G, M, X_1, \ldots, X_k)$**.**

Input: G is an n-vertex k-hypergraph to which `VerifyGuess` has (only) coloured oracle access. n and M are positive powers of two, and $X_1, \ldots, X_k \subseteq V(G)$ are disjoint.

Behaviour: Let $p_{\mathsf{out}} = (8k \log n)^{-k}$.

Completeness: If $e(G[X_1, \ldots, X_k]) \geq M$, then `VerifyGuess` outputs `Yes` with probability at least p_{out}.

Soundness: If $e(G[X_1, \ldots, X_k]) < M \cdot p_{\mathsf{out}}/2(k \log n)^k$, then `VerifyGuess` outputs `Yes` with probability at most $p_{\mathsf{out}}/2$.

(C1) For each $i \in [k]$ and each $0 \leq j \leq k \log n$, construct a subset $Y_{i,j}$ of X_i by including each vertex independently with probability $1/2^j$. Construct the finite set A of all tuples $(a_1, \ldots, a_k)$ with $0 \leq a_1, \ldots, a_k \leq k \log n$ and $a_1 + \cdots + a_k \geq \log M$.

(C2) For each tuple $(a_1, \ldots, a_k) \in A$: If $\mathrm{cIND}_G(Y_{1,a_1}, \ldots, Y_{k,a_k}) = 0$, then halt and output `Yes`.

(C3) We have not halted yet, but do so now and output `No`.

4.2. **Solving the gap problem.**

Lemma 21. `VerifyGuess` *behaves as stated, runs in time* $\mathcal{O}(nk^k \log^k n)$*, and makes at most* $\mathcal{O}(k^k \log^k n)$ *oracle queries.*

Proof. Let $G, M, X_1, \ldots, X_k$ be the input for `VerifyGuess`, and let $H = G[X_1, \ldots, X_k]$. For notational convenience, we denote the gap in the soundness case by γ, that is, we set $\gamma := p_{\mathsf{out}}/2(k \log n)^k = 1/2^{3k+1} k^{2k} \log^{2k} n$.

Running time and oracle queries. Step (C1) takes $\mathcal{O}(nk \log n + k^k \log^k n)$ time and no oracle queries; step (C2) takes $\mathcal{O}(k^k \log^k n)$ time and $\mathcal{O}(k^k \log^k n)$ oracle queries; and step (C3) takes $\mathcal{O}(1)$ time and no oracle queries. The claimed bounds follow, and it remains to prove that the soundness and completeness properties hold.

Soundness. We next prove soundness, as this is the easier part of proving correctness. So suppose $e(H) \leq \gamma M$. Let $(a_1, \ldots, a_k) \in A$, and let $H' \subseteq H$ denote the random induced subgraph $G[Y_{1,a_1}, \ldots, Y_{k,a_k}]$. Then for all $e \in E(H)$, we have

$$\mathbb{P}(e \in E(H')) = \prod_{j=1}^{k} 2^{-a_j} \leq \frac{1}{M} \leq \frac{\gamma}{e(H)} \leq \frac{p_{\mathsf{out}}}{2|A|e(H)}.$$

By a union bound over all $e \in E(H)$ and all $(a_1, \ldots, a_k) \in A$, it follows that the probability that `VerifyGuess` outputs `Yes` is at most $p_{\mathsf{out}}/2$. This establishes the soundness of the algorithm, so it remains to prove completeness.

Completeness. Suppose now that $e(H) \geq M$ holds. We must prove that `VerifyGuess` outputs `Yes` with probability at least p_{out}. It suffices to show that with probability at least p_{out}, there is at least one setting of the vector $(a_1, \ldots, a_k) \in A$ such that $G[Y_{1,a_1}, \ldots, Y_{k,a_k}]$ contains at least one edge.

We will define this setting iteratively. First, with reasonable probability, we will find an integer a_1 and a vertex $v_1 \in Y_{1,a_1}$ such that $G[v_1, X_2, \ldots, X_k]$ contains roughly $2^{-a_1} e(H)$ edges. In the process, we expose Y_{1,a_1}. We then, again with reasonable probability, find an integer a_2 and a vertex $v_2 \in Y_{2,a_2}$ such that $G[v_1, v_2, X_3, \ldots, X_k]$ contains roughly $2^{-a_1-a_2} e(H)$ edges. Continuing in this vein, we eventually find $(a_1, \ldots, a_k) \in A$ and vertices $v_i \in Y_{i,a_i}$ such that $\{v_1, \ldots, v_k\}$ is an edge in $G[Y_{1,a_1}, \ldots, Y_{k,a_k}]$, proving the result.

More formally, for all $i \in [k]$, let $\mathcal{E}_i$ be the event that there exist $0 \leq a_1, \ldots, a_i \leq k \log n$ and $v_1, \ldots, v_i \in V(H)$ such that:

(a) for all $j \in [i]$, $v_j \in Y_{j,a_j}$;
(b) we have $d_H(v_1, \ldots, v_i) \geq e(H)/\prod_{j=1}^{i} 2^{a_j}$.

We make the following **Claim**: $\mathbb{P}(\mathcal{E}_1) \geq 1/(8k \log n)$ and, for all $2 \leq i \leq k$, $\mathbb{P}(\mathcal{E}_i \mid \mathcal{E}_{i-1}) \geq 1/(8k \log n)$.

Proof of Lemma 21 from Claim: Suppose $\mathcal{E}_k$ occurs, and let $a_1, \ldots, a_k$ and $v_1, \ldots, v_k$ be as in the definition of $\mathcal{E}_k$. By (b), $d_H(v_1, \ldots, v_k) > 0$, so $\{v_1, \ldots, v_k\}$ is an edge in H; it follows by (a) that it is also an edge in $G[Y_{1,a_1}, \ldots, Y_{k,a_k}]$. Also by (b), since $d_H(v_1, \ldots, v_k) = 1$, we have $\prod_{j=1}^{k} 2^{a_j} \geq e(H) \geq M$, so $a_1 + \cdots + a_k \geq \log M$. Thus $(a_1, \ldots, a_k) \in A$, so whenever $\mathcal{E}_k$ occurs, `VerifyGuess` outputs `Yes` on reaching $(a_1, \ldots, a_k)$ in step (C2). By the Claim, we have

$$\mathbb{P}(\mathcal{E}_k) = \mathbb{P}(\mathcal{E}_1) \prod_{j=2}^{k} \mathbb{P}(\mathcal{E}_j \mid \mathcal{E}_1, \ldots, \mathcal{E}_{j-1}) = \mathbb{P}(\mathcal{E}_1) \prod_{j=2}^{k} \mathbb{P}(\mathcal{E}_j \mid \mathcal{E}_{j-1}) \geq 1/(8k \log n)^k = p_{\mathsf{out}},$$

so completeness follows and hence so does the lemma statement.

Proof of Claim: We first prove the claim for $\mathcal{E}_1$. We will choose a_1 depending on the degree distribution of vertices in X_1. For all integers $1 \leq d \leq k \log n$, let

$$X_1^d := \{v \in X_1 \colon 2^{d-1} \leq d_H(v) < 2^d\}$$

be the set of vertices in X_1 with degree roughly 2^d. Every edge in H is incident to exactly one vertex in exactly one set X_1^d, so there exists D such that X_1^D is incident to at least $e(H)/k \log n$ edges of H. We take $a_1 := \lceil \log e(H) \rceil - D + 1$. Note that $0 \leq a_1 \leq k \log n$, since $X_1^D \neq \emptyset$ and so $e(H) \geq 2^{D-1}$.

We would like to take $v_1 \in Y_{1,a_1} \cap X_1^D$, so we next bound the probability that this set is non-empty. We have

$$\mathbb{P}(X_1^D \cap Y_{1,a_1} \neq \emptyset) = 1 - (1 - 2^{-a_1})^{|X_1^D|} \geq 1 - \exp(-2^{-a_1}|X_1^D|).$$

Since every vertex in X_1^D has degree at most 2^D, by the definition of D we have $2^D|X_1^D| \geq e(H)/(k \log n)$. Moreover, we have $a_1 \leq \log e(H) - D + 2$. It follows that

$$\mathbb{P}(X_1^D \cap Y_{1,a_1} \neq \emptyset) \geq 1 - \exp\left(-\frac{2^{D-2}}{e(H)} \cdot \frac{e(H)}{k 2^D \log n}\right) = 1 - \exp\left(-\frac{1}{4k \log n}\right) \geq \frac{1}{8k \log n}.$$

Suppose $X_1^D \cap Y_{1,a_1} \neq \emptyset$, and take $v_1 \in X_1^D \cap Y_{1,a_1}$. Then v_1 certainly satisfies (a), and by the definitions of a_1 and X_1^D we have $e(H)/2^{a_1} \leq 2^{D-1} \leq d_H(v_1)$, so v_1 also satisfies (b). We have therefore shown $\mathbb{P}(\mathcal{E}_1) \geq 1/(8k \log n)$ as required.

Now let $2 \leq i \leq k$. The argument is similar, but we include it explicitly for the benefit of the reader. We first expose $Y_{1,a_1}, \ldots, Y_{i-1,a_{i-1}}$: Let $\mathcal{F}$ be a possible filtration of these variables consistent with $\mathcal{E}_{i-1}$, and let $a_1, \ldots, a_{i-1}$ and $v_1, \ldots, v_{i-1}$ be as in the definition of $\mathcal{E}_{i-1}$. It then suffices to show that $\mathbb{P}(\mathcal{E}_i \mid \mathcal{F}) \geq 1/(8k \log n)$.

Similarly to the $i = 1$ case, for all integers $1 \leq d \leq k \log n$, let

$$X_i^d := \{v \in X_i \colon 2^{d-1} \leq d_H(v_1, \ldots, v_{i-1}, v) < 2^d\}.$$

Every edge in $H[v_1, \ldots, v_{i-1}, X_i, \ldots, X_k]$ is incident to exactly one vertex in exactly one set X_i^d, so there exists D_i such that $X_i^{D_i}$ is incident to at least $d_H(v_1, \ldots, v_{i-1})/k \log n$ edges of $H[v_1, \ldots, v_{i-1}, X_i, \ldots, X_k]$. We take $a_i := \lceil \log d_H(v_1, \ldots, v_{i-1}) \rceil - D_i + 1$; note that $0 \leq a_i \leq k \log n$.

As in the $i = 1$ case, we would like to take $v_i \in Y_{i,a_i} \cap X_i^{D_i}$, so we next bound the probability that this set is non-empty. Since every vertex $v \in X_i^{D_i}$ satisfies $d_H(v_1, \ldots, v_{i-1}, v) \leq 2^{D_i}$, we have $2^{D_i}|X_i^{D_i}| \geq$

$d_H(v_1,\dots,v_{i-1})/k\log n$. It follows that

$$\begin{aligned}\mathbb{P}(X_i^{D_i}\cap Y_{i,a_i}\neq\emptyset\mid\mathcal{F}) &= 1-(1-2^{-a_i})^{|X_i^{D_i}|}\geq 1-\exp(-2^{-a_i}|X_i^{D_i}|)\\ &\geq 1-\exp\Big(-\frac{2^{D_i-2}}{d_H(v_1,\dots,v_{i-1})}\cdot\frac{d_H(v_1,\dots,v_{i-1})}{k2^{D_i}\log n}\Big)\\ &= 1-\exp\Big(-\frac{1}{4k\log n}\Big)\geq\frac{1}{8k\log n}.\end{aligned}$$

Suppose $X_i^{D_i}\cap Y_{i,a_i}\neq\emptyset$, and take $v_i\in X_i^{D_i}\cap Y_{i,a_i}$. Then v_i certainly satisfies (a). By the definitions of a_i and $X_i^{D_i}$, and the fact that $v_1,\dots,v_{i-1}$ satisfy (b), we have

$$e(H)/\prod_{j=1}^{i}2^{a_j}\leq d_H(v_1,\dots,v_{i-1})/2^{a_i}\leq 2^{D_i-1}\leq d_H(v_1,\dots,v_i).$$

Thus (b) is satisfied, and we have shown $\mathbb{P}(\mathcal{E}_i\mid\mathcal{F})\geq 1/(8k\log n)$ as required. □

4.3. **Proving Lemma 18.** We next turn `VerifyGuess` into a crude approximation algorithm for k-partite k-hypergraphs in the natural way.

Algorithm `ColourCoarse`$(G,X_1,\dots,X_k)$**.**
Input: G is an n-vertex k-hypergraph, where n is a power of two, to which `ColourCoarse` has coloured oracle access (only). $X_1,\dots,X_k$ form a partition of $V(G)$.
Behaviour: Let $b:=(4k\log n)^k$. Then `ColourCoarse` (G) outputs a non-negative integer m such that, with probability at least $2/3$, $m/b\leq e(G[X_1,\dots,X_k])\leq mb$.

(D1) Set $p_{\mathsf{out}}:=(8k\log n)^{-k}$ and $N:=\lceil 48\ln(6k\log n)/p_{\mathsf{out}}\rceil$.
(D2) For each M in $\{1,2,4,8,\dots,n^k\}$: Execute `VerifyGuess` $(G,M,X_1,\dots,X_k)$ a total of N times, and let $S_M\in\{0,\dots,N\}$ be the number of executions that returned `Yes`. (Naturally we use independent randomness for each value of M.)
(D3) If $\mathrm{cIND}_G(X_1,\dots,X_k)=1$, let $m=0$. Otherwise, if there exists M such that $S_M\geq\frac{3}{4}p_{\mathsf{out}}N$, let m be the least such M. Otherwise, let $m=n^k$. Output $m\big(p_{\mathsf{out}}/2k^k\log^k n\big)^{1/2}$.

Lemma 22. `ColourCoarse` *behaves as stated, runs in time* $\mathcal{O}((8k\log n)^{2k+2}n)$*, and requires* $\mathcal{O}((8k\log n)^{2k+2})$ *oracle queries.*

Proof. Let G and $X_1,\dots,X_k$ be the inputs, so that G is an n-vertex k-hypergraph and $X_1,\dots,X_k$ partition $V(G)$.

Running time and oracle queries. Observe $N=\mathcal{O}((8k\log n)^{k+1})$. `ColourCoarse` just executes `VerifyGuess` at most $\mathcal{O}(\log(n^k)N)$ times. By Lemma 21, each execution takes $\mathcal{O}(nk^k\log^k n)$ time and makes $\mathcal{O}(k^k\log^k n)$ oracle queries. Thus the claimed bounds on the running time and number of oracle queries of `ColourCoarse` follow.

Correctness. Let $M\in\{1,2,4,8,\dots,n^k\}$, and let $H=G[X_1,\dots,X_k]$. For this fixed M, the algorithm invokes `VerifyGuess` N times, so the random variable S_M is the sum of N independent indicator variables. By a standard Chernoff bound (Lemma 13(i) taking $\varepsilon=1/4$),

$$\mathbb{P}\big(|S_M-\mathbb{E}(S_M)|\geq\mathbb{E}(S_M)/4\big)\leq 2\exp\big(-\tfrac{1}{48}\mathbb{E}(S_M)\big)\tag{11}$$

If $e(H)\geq M$, then the completeness of `VerifyGuess` implies $\mathbb{E}(S_M)\geq Np_{\mathsf{out}}$. Thus (11) and our choice of N imply that

$$\mathbb{P}\big(S_M<\tfrac{3}{4}Np_{\mathsf{out}}\big)\leq 2\exp\big(-\tfrac{1}{48}Np_{\mathsf{out}}\big)\leq 1/(3k\log n).$$

Similarly, if $M > 2(k\log n)^k \cdot e(H)/p_{\mathsf{out}}$, then the soundness of `VerifyGuess` implies $\mathbb{E}(S_M) \leq Np_{\mathsf{out}}/2$. But then (11) implies that

$$\mathbb{P}\big(S_M > \tfrac{3}{4}Np_{\mathsf{out}}\big) \leq \mathbb{P}\big(S_M > \tfrac{5}{4}\mathbb{E}(S_M)\big) \leq 2\exp\big(-\tfrac{1}{48}Np_{\mathsf{out}}\big) \leq 1/(3k\log n).$$

Finally, we perform a union bound over all $M \in \{1, 2, 4, 8, \ldots, n^k\}$ that satisfy either $M \leq e(H)$ or $M \geq 2k^k \log^k n \cdot e(H)/p_{\mathsf{out}}$. (Note that no value of M satisfies both inequalities.) There are at most $k\log n$ such M's, so with probability at least $2/3$, we see $S_M < 3Np_{\mathsf{out}}/4$ for all $M \leq e(H)$ and $S_M > 3Np_{\mathsf{out}}/4$ for all $M \geq 2k^k \log^k n \cdot e(H)/p_{\mathsf{out}}$. By the definition of m, it follows that in this case

$$\frac{p_{\mathsf{out}}}{2k^k \log^k n} m \leq e(H) \leq m.$$

Hence, writing $x = m(p_{\mathsf{out}}/2k^k \log^k n)^{1/2}$ for the output of `ColourCoarse`,

$$x\sqrt{\frac{p_{\mathsf{out}}}{2k^k \log^k n}} \leq e(H) \leq x\Big/\sqrt{\frac{p_{\mathsf{out}}}{2k^k \log^k n}}.$$

Since $p_{\mathsf{out}} = 1/(8k\log n)^k$, the output of `ColourCoarse` approximates $e(H)$ up to a factor of $(4k\log n)^k$ as required. □

We now combine our algorithm for coarsely approximately counting edges in k-partite k-hypergraphs with colour-coding to obtain an algorithm for general k-hypergraphs.

Algorithm `HelperCoarse`(G)**.**

Input: G is an n-vertex k-hypergraph, where n is a power of two, to which `HelperCoarse` has coloured oracle access (only).

Behaviour: `HelperCoarse`(G) outputs a non-negative integer $\hat{e}$ which, with probability at least $2/3$, satisfies $\hat{e}/2(4k\log n)^k \leq e(G) \leq \hat{e} \cdot 2(4k\log n)^k$.

(E1) Let $t = 3e^{2k}$, and let $T = \lceil 72\ln t\rceil + 3$.
(E2) For each $i \in [t]$:
  (E3) Sample a uniformly random function $c_i : V(G) \to [k]$, which yields a random k-partition $X_1, \ldots, X_k$ of $V(G)$.
  (E4) Execute `ColourCoarse`$(G, X_1, \ldots, X_k)$ exactly T times and let M_i be the median output produced by these executions.
(E3) Output $\frac{k^k}{tk!}\sum_{i=1}^{t} M_i$.

Lemma 23. `HelperCoarse` *behaves as stated, runs in time* $\mathcal{O}(k^{3k} n \log^{2k+2} n)$*, and requires* $\mathcal{O}(k^{3k}\log^{2k+2} n)$ *oracle queries.*

Proof. Let G be an n-vertex k-hypergraph input for `HelperCoarse`.

Running time and oracle queries. It is clear that the bottleneck in both the running time and the number of oracle queries is the Tt total invocations of `ColourCoarse`. Recall from Lemma 22 that each such invocation runs in time $\mathcal{O}(n(8k\log n)^{2k+2})$ and requires $\mathcal{O}((8k\log n)^{2k+2})$ oracle queries. Since $t = \mathcal{O}(e^{2k})$ and $T = \mathcal{O}(k)$, the claimed bounds follow.

Correctness. Let $b := (4k\log n)^k$ be the approximation ratio of `ColourCoarse`. For all $i \in [t]$, let $G_i = G[c_i^{-1}(1), \ldots, c_i^{-1}(k)]$ be the ith hypergraph we consider, and let $m_i = e(G_i)$. Let $x_{i,j}$ be the output of the jth call to `ColourCoarse` in evaluating M_i, and let $\mathcal{E}_{i,j}$ be the event that $x_{i,j}/b \leq m_i \leq x_{i,j}b$.

Note that the $\mathcal{E}_{i,j}$'s are independent conditioned on c_i, and that the correctness of `ColourCoarse` (Lemma 22) implies that $\mathbb{P}(\mathcal{E}_{i,j}) \geq 2/3$ for all $j \in [T]$. Moreover, for all $i \in [t]$, if at least half the $\mathcal{E}_{i,j}$'s

occur, then $M_i/b \le m_i \le bM_i$. Thus by a Chernoff bound (Lemma 13(i) applied with $\varepsilon = 1/4$ and $\mu = 2T/3$), we have

$$\mathbb{P}(\tfrac{1}{b}M_i \le m_i \le bM_i \mid c_i) \ge 1 - 2e^{-T/72} \ge 1 - 2e^{-\ln t - 3} > 1 - 1/(6t).$$

It follows by a union bound that, with probability at least $5/6$, $M_i/b \le m_i \le bM_i$ for all $i \in [t]$.

Now observe that $\mathbb{E}(\sum_i m_i) = t(k!/k^k)e(G)$, and that each m_i lies in $[0, e(G)]$. It follows by Hoeffding's inequality (Lemma 11) that

$$\begin{aligned}\mathbb{P}\Big(\Big|\frac{k^k}{tk!}\sum_i m_i - e(G)\Big| > \frac{1}{2}e(G)\Big) &= \mathbb{P}\Big(\Big|\sum_i m_i - \frac{tk!\cdot e(G)}{k^k}\Big| > \frac{tk!}{2k^k}e(G)\Big)\\ &\le 2\exp\left(-2\Big(\frac{tk!}{k^k}e(G)\Big)^2/te(G)^2\right) = 2\exp\left(-2t(k!/k^k)^2\right).\end{aligned}$$

By Stirling's formula and our definition of t, it follows that

$$\mathbb{P}\Big(\Big|\frac{k^k}{tk!}\sum_i m_i - e(G)\Big| > \frac{1}{2}e(G)\Big) \le 2\exp\left(-te^{-2k}\right) \le 1/6.$$

Thus with probability at least $5/6$, we have $e(G)/2 \le \frac{k^k}{tk!}\sum_i m_i \le 2e(G)$.

It now follows by a union bound that with probability at least $2/3$, $\frac{1}{2b}\cdot e(G) \le \frac{k^k}{tk!}\sum_i M_i \le 2b\cdot e(G)$ as required. □

Lemma 18 now follows via the usual probability amplification argument.

Lemma 18 (restated)**.** *There is a randomised algorithm* `Coarse`(G,δ) *with the following behaviour. Suppose G is an n-vertex k-hypergraph to which* `Coarse` *has (only) coloured oracle access, where n is a power of two, and suppose $0 < \delta < 1$. Then in time $\mathcal{O}(\log(1/\delta)k^{3k}n\log^{2k+2}n)$, and using at most $\mathcal{O}(\log(1/\delta)k^{3k}\log^{2k+2}n)$ queries to* cIND$_G$, `Coarse`(G,δ) *outputs a rational number $\hat{e}$. Moreover, with probability at least $1-\delta$,*

$$\frac{e(G)}{2(4k\log n)^k} \le \hat{e} \le e(G)\cdot 2(4k\log n)^k.$$

Proof. Given G and $\delta > 0$, we simply invoke `HelperCoarse`(G) a total of $T := \lceil 36\ln(2/\delta)\rceil$ times and return the median output. By the correctness of `HelperCoarse` (Lemma 23), each invocation returns a valid approximation of $e(G)$ with probability at least $2/3$, and if at least $T/2$ invocations return valid approximations then the median is also a valid approximation. It follows by Chernoff bounds (Lemma 13(i) with $\varepsilon = 1/2$ and $\mu = T/3$) that we output a valid approximation with probability at least $1 - 2e^{-T/36} \ge 1-\delta$, as required, and our bounds on running time and oracle usage are immediate from Lemma 23. □

5. Approximately uniform sampling

In this section we demonstrate that we can use our approximate counting algorithm to sample an edge almost uniformly at random, proving Theorem 2. The core of our algorithm is the following subroutine.

Algorithm `HelperSample`(G,ε)**.**

Input: G is an n-vertex k-hypergraph containing at least one edge, where n is a power of two, to which `HelperCoarse` has coloured oracle access (only). $0 < \varepsilon < 1/2$ is a rational number.

Behaviour: With probability at least $1 - \varepsilon/n^k$, `HelperSample`(G,ε) outputs a sample from a distribution $\hat{U}$ on $E(G)$ such that, for all $e \in E(G)$, $\hat{U}(e) \in (1\pm\varepsilon)/e(G)$.

(S1) If $\varepsilon \le n^{-k}$, then enumerate the edges of G using $\binom{n}{k}$ invocations of cIND$_G$ and return a uniformly-sampled edge.

(S2) Let $I = \log n - \lceil \log(8k^2) \rceil$, $\xi = \varepsilon/(100 \log n)$, and $\delta = \xi/2^{k+8}n^{2k}$. If $I \leq 1$, enumerate the edges of $e(G)$ using cIND_G and return a uniformly random sample.

(S3) Let $X_1 \leftarrow V(G)$, $M_1 \leftarrow \texttt{Count}(X_1, \xi, \delta)$, and $i \leftarrow 2$. While $i \leq I$:

- (a) Choose a size-$(|X_{i-1}|/2)$ set $X \subseteq X_{i-1}$ uniformly at random, and let $M \leftarrow \texttt{Count}(G[X], \xi, \delta)$.
- (b) If $M_{i-1} = 0$, output `Fail`. Otherwise, with probability $\max\{0, 1 - M/M_{i-1}\}$, go to (a) (i.e. reject X and resample).
- (c) Accept X by setting $X_i \leftarrow X$, $M_i \leftarrow M$ and $i \leftarrow i + 1$.

(S4) Enumerate the edges of $G[X_I]$ using cIND_G and return a uniformly random sample.

Lemma 24. $\texttt{HelperSample}(G, \varepsilon)$ *behaves as claimed. With probability at least* $1 - \varepsilon/n^k$*, writing* $T = \varepsilon^{-2}k^{7k}\log^{4k+11} n$*, its running time is* $\mathcal{O}(nT)$*, and it invokes* cIND_G *at most* $\mathcal{O}(T)$ *times.*

Proof. If $\varepsilon \leq n^{-k}$ or $I \leq 1$, then both correctness and the stated time bounds are clear, so suppose $\varepsilon > n^{-k}$ and $I \geq 2$ (which implies $n \geq 32k^2$). We first carefully bound the probability that something goes wrong over the course of the algorithm's execution.

For all $r \in [I]$, let $\mathcal{E}_r$ be the event that `Count` is called at most $2^{k+2}\ln(8In^k/\varepsilon)$ times in calculating M_r, that each time it is called it returns M satisfying $M \in (1 \pm \xi)e(G[X])$, and that $M_r > 0$. (Intuitively, $\mathcal{E}_r$ is the event that the algorithm behaves as we expect in determining X_r.) We will bound $\mathbb{P}(\mathcal{E}_r \mid \mathcal{E}_1, \ldots, \mathcal{E}_{r-1})$ below for all $r \in [I]$, and hence bound $\mathbb{P}(\mathcal{E}_1, \ldots, \mathcal{E}_r)$ below. When $r = 1$, it follows from Theorem 1 and the fact that $e(G) > 0$ that $\mathbb{P}(\mathcal{E}_1) \geq 1 - \delta$.

Let $2 \leq r \leq I$, let $\mathcal{F}$ be a possible filtration of $X_1, \ldots, X_{r-1}$ and $M_1, \ldots, M_{r-1}$ compatible with $\mathcal{E}_1, \ldots, \mathcal{E}_{r-1}$, and let Y be the value of X_{r-1} determined by $\mathcal{F}$. Let $\mathcal{E}_{r,1}$ be the event that when $i = r$, we accept a set X_r within $2^{k+2}\ln(8In^k/\varepsilon)$ iterations of (S3a)–(S3c). Let $\mathcal{E}_{r,2}$ be the event that when $i = r$, we accept a set X_r without `Count` ever returning an inaccurate estimate $M \notin (1 \pm \xi)e(G[X])$. Note that if $\mathcal{E}_{r,2} \cap \mathcal{F}$ occurs, then $M_r \in (1 \pm \xi)e(G[X_r])$; moreover, since we accepted X_r with probability at most M_r/M_{r-1}, we must have $M_r > 0$. Thus

$$\mathbb{P}(\mathcal{E}_r \mid \mathcal{F}) \geq \mathbb{P}(\mathcal{E}_{r,1} \cap \mathcal{E}_{r,2} \mid \mathcal{F}). \tag{12}$$

To bound $\mathbb{P}(\mathcal{E}_{r,1} \cap \mathcal{E}_{r,2} \mid \mathcal{F})$ below, consider the first iteration of (S3a)–(S3c) with $i = r$. Let $\mathcal{A}_1$ be the event that `Count` returns an inaccurate estimate M, and let $\mathcal{A}_2$ be the event that `Count` returns an accurate estimate M and we subsequently accept X. Each of these events is independent of past samples, so

$$\mathbb{P}(\mathcal{E}_{r,1} \mid \mathcal{F}) \geq 1 - \big(1 - \mathbb{P}(\mathcal{A}_2 \mid \mathcal{F})\big)^{2^{k+2}\ln(8In^k/\varepsilon)}, \tag{13}$$

$$\mathbb{P}(\mathcal{E}_{r,2} \mid \mathcal{F}) = \frac{\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F})}{\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F}) + \mathbb{P}(\mathcal{A}_1 \mid \mathcal{F})}. \tag{14}$$

(Here (14) follows on exposing the number T of iterations of (S3a)–(S3c) before either `Count` returns an inaccurate estimate or we accept an accurate estimate; by Bayes' theorem we have $\mathbb{P}(\mathcal{E}_{r,2} \mid \mathcal{F} \text{ and } T = t) = \mathbb{P}(\mathcal{A}_2 \mid \mathcal{F})/(\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F}) + \mathbb{P}(\mathcal{A}_1 \mid \mathcal{F}))$ for all $t \geq 0$.)

We now bound $\mathbb{P}(\mathcal{A}_1 \mid \mathcal{F})$ above and $\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F})$ below. Theorem 1 implies that

$$\mathbb{P}(\mathcal{A}_1 \mid \mathcal{F}) \leq \delta. \tag{15}$$

Moreover, for all $S \subset Y$ with $|S| = |Y|/2$, Theorem 1 implies that

$$\mathbb{P}\big(\mathcal{A}_2 \text{ occurs and } X = S \mid \mathcal{F}\big) \geq \binom{|Y|}{|Y|/2}^{-1} \cdot (1 - \delta) \cdot \frac{(1 - \xi)e(G[S])}{M_{r-1}}.$$

By the definition of $\mathcal{F}$ we have $M_{r-1} \in (1 \pm \xi)e(G[Y])$, so it follows that

$$\mathbb{P}\big(\mathcal{A}_2 \text{ occurs and } X = S \mid \mathcal{F}\big) \geq \frac{1}{2}\binom{|Y|}{|Y|/2}^{-1} \frac{e(G[S])}{e(G[Y])}.$$

On summing both sides over S, since each edge of $G[Y]$ appears in exactly $\binom{|Y|-k}{|Y|/2-k}$ sets $S \subset Y$ with $|S| = |Y|/2$, we obtain

$$\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F}) \geq \frac{1}{2}\binom{|Y|}{|Y|/2}^{-1}\binom{|Y|-k}{|Y|/2-k}.$$

Since $r \leq I$, we have $|Y| \geq n/2^{I-1} \geq 4k^2$, so by Lemma 14 it follows that $\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F}) \geq 2^{-k-2}$. It therefore follows from (13), (14) and (15) that

$$\mathbb{P}(\mathcal{E}_{r,1} \mid \mathcal{F}) \geq 1 - (1-2^{-k-2})^{2^{k+2}\ln(8In^k/\varepsilon)} \geq 1 - e^{-\ln(8In^k/\varepsilon)} = 1 - \varepsilon/8In^k,$$
$$\mathbb{P}(\mathcal{E}_{r,2} \mid \mathcal{F}) \geq 2^{-k-2}/(2^{-k-2}+\delta) \geq 1 - 2^{k+2}\delta \geq 1 - \varepsilon/8In^k.$$

By (12), it follows that $\mathbb{P}(\mathcal{E}_r \mid \mathcal{F}) \geq 1 - \varepsilon/4In^k$ for all $r \in [I]$. Thus

$$\mathbb{P}(\mathcal{E}_1, \ldots, \mathcal{E}_I) \geq 1 - \varepsilon/4n^k. \tag{16}$$

With (16) in hand, we are now ready to proceed with the main proof.

Running time and oracle queries. Since $|X_I| = n/2^{I-1} \leq 32k^2$ by our choice of I, the bottleneck in the running time is the invocations of `Count` in (S3b). By Theorem 1, each invocation takes time $\mathcal{O}(\log(1/\delta)\xi^{-2}k^{6k}n\log^{4k+7}n)$ and requires $\mathcal{O}(\log(1/\delta)\xi^{-2}k^{6k}\log^{4k+7}n)$ invocations of the oracle. Since $1/\xi = \mathcal{O}(\log n/\varepsilon)$, $\log(1/\delta) = \mathcal{O}(k\log n + \log(1/\varepsilon))$ and $1/\varepsilon = \mathcal{O}(n^k)$, each invocation takes time $\mathcal{O}(\varepsilon^{-2}k^{6k+1}n\log^{4k+10}n)$ and requires $\mathcal{O}(\varepsilon^{-2}k^{6k+1}\log^{4k+10}n)$ oracle invocations. By (16), with probability at least $1-\varepsilon/n^k$, there are at most $2^{k+2}\ln(8In^k/\varepsilon) = \mathcal{O}(k2^k\log n)$ such invocations, so the claimed bounds follow.

Correctness. Let F be the output of `HelperSample`(G,ε), or `Fail` if `HelperSample`(G,ε) does not halt. Since $e(G) > 0$, by (16), `HelperSample`(G,ε) outputs a sample from $E(G)$ with probability at least $1-\varepsilon/n^k$; thus to prove Lemma 24, it suffices to show that for all $f \in E(G)$ we have $\mathbb{P}(F = f) \in (1 \pm \varepsilon)/e(G)$.

Let $S_1 = V(G)$ and, for all $S_1 \supset S_2 \supset \cdots \supset S_I \supset f$ with $|S_r| = n/2^{r-1}$ for all $r \in [I]$, let

$$p(S_1, \ldots, S_I, f) = \mathbb{P}(X_r = S_r \text{ for all } r \in [I],\ F = f, \text{ and } \mathcal{E}_1, \ldots, \mathcal{E}_I \text{ occur}).$$

Thus for all $f \in E(G)$, we have

$$\mathbb{P}(F = f) \geq \sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} p(S_1, \ldots, S_I, f),$$
$$\mathbb{P}(F = f) \leq \sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} p(S_1, \ldots, S_I, f) + \big(1 - \mathbb{P}(\mathcal{E}_1 \cap \cdots \cap \mathcal{E}_I)\big).$$

By (16), it follows that

$$\sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} p(S_1, \ldots, S_I, f) \leq \mathbb{P}(F = f) \leq \sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} p(S_1, \ldots, S_I, f) + \frac{\varepsilon}{4n^k}. \tag{17}$$

To bound each term $p(S_1, \ldots, S_I, f)$, we first derive estimates for the probability that $X_r = S_r$, conditioned on $X_t = S_t$ for all $t \in [r-1]$ and on $\mathcal{E}_1, \ldots, \mathcal{E}_{r-1}$. Let $\mathcal{F}_r$ be an arbitrary filtration of $X_1, \ldots, X_{r-1}$ and $M_1, \ldots, M_{r-1}$ compatible with these events. For all $S \subset S_{r-1}$ with $|S| = |S_{r-1}|/2$, let $q(S)$ be the probability that we accept S on a given iteration of (S3a)–(S3c) conditioned on $X = S$ and $\mathcal{F}_r$. Then by Theorem 1 and the definition of $\mathcal{F}_r$,

$$q(S) \geq (1-\delta)\frac{(1-\xi)e(G[S])}{(1+\xi)e(G[S_{r-1}])} \text{ and } q(S) \leq \frac{(1+\xi)e(G[S])}{(1-\xi)e(G[S_{r-1}])} + \delta. \tag{18}$$

Moreover, by a standard rejection sampling argument (see e.g. Florescu [20, Proposition 3.3]), we have

$$\mathbb{P}(X_r = S_r \mid \mathcal{F}_r) = \frac{q(S_r)}{\sum_{\substack{T \subset S_{r-1} \\ |T| = |S_{r-1}|/2}} q(T)}. \tag{19}$$

By (18) and (19), and using the fact that $e(G[S_r]) \geq 1$, we have

$$\mathbb{P}(X_r = S_r \mid \mathcal{F}_r) \leq \frac{(1+\xi)^2}{(1-\xi)^2(1-\delta)} \cdot \frac{e(G[S_r]) + \delta e(G[S_{r-1}])}{\sum_{\substack{T \subset S_{r-1} \\ |T| = |S_{r-1}|/2}} e(G[T])} \leq \frac{(1+\xi)^3}{(1-\xi)^3} \cdot \frac{e(G[S_r])}{\sum_{\substack{T \subset S_{r-1} \\ |T| = |S_{r-1}|/2}} e(G[T])}.$$

Since each edge of $G[S_{r-1}]$ appears exactly $\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k}$ times in this sum, it follows that

$$\mathbb{P}(X_r = S_r \mid \mathcal{F}_r) \leq \frac{(1+\xi)^3}{(1-\xi)^3} \cdot \frac{e(G[S_r])}{\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k} e(G[S_{r-1}])} \leq (1+8\xi) \frac{e(G[S_r])}{\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k} e(G[S_{r-1}])}. \tag{20}$$

(Here the last inequality follows since $\xi < 1/20$.)

Also by (18) and (19), we have

$$\begin{aligned}\mathbb{P}(X_r = S_r \mid \mathcal{F}_r) &\geq \frac{(1-\delta)(1-\xi)^2}{(1+\xi)^2} \cdot \frac{e(G[S_r])}{\sum_{\substack{T \subset S_{r-1} \\ |T| = |S_{r-1}|/2}} \big(e(G[T]) + \delta e(G[S_{r-1}])\big)} \\ &\geq \frac{(1-\xi)^3}{(1+\xi)^2} \cdot \frac{e(G[S_r])}{\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k} e(G[S_{r-1}]) + \binom{|S_{r-1}|}{|S_{r-1}|/2} \delta e(G[S_{r-1}])}.\end{aligned}$$

Since $|S_{r-1}| = n/2^{r-1} \geq 4k^2$, by Lemma 14 we have

$$\binom{|S_{r-1}|}{|S_{r-1}|/2} \leq 2^{k+1} \binom{|S_{r-1}| - k}{|S_{r-1}|/2 - k}.$$

It follows that

$$\begin{aligned}\mathbb{P}(X_r = S_r \mid \mathcal{F}_r) &\geq \frac{(1-\xi)^3}{(1+\xi)^2} \cdot \frac{e(G[S_r])}{(1+2^{k+1}\delta)\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k} e(G[S_{r-1}])} \\ &\geq (1-6\xi) \frac{e(G[S_r])}{\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k} e(G[S_{r-1}])}. \end{aligned} \tag{21}$$

With upper and lower bounds on $\mathbb{P}(X_r = S_r \mid \mathcal{F}_r)$ now in place, we return to the task of bounding $p(S_1, \ldots, S_I, f)$. Observe that for all $f \in E(G)$,

$$\sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} p(S_1, \ldots, S_I, f) = \sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} \frac{1}{e(G[S_I])} \prod_{r=1}^{I} \mathbb{P}\big(X_r = S_r \text{ and } \mathcal{E}_r \text{ occurs} \mid \mathcal{F}_r\big).$$

It therefore follows from (20) that

$$\sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} p(S_1, \ldots, S_I, f) \leq \sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} \frac{1}{e(G[S_I])} \prod_{r=2}^{I} \frac{(1+8\xi) e(G[S_r])}{\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k} e(G[S_{r-1}])}.$$

We have $(1+8\xi)^I \leq e^{8I\xi} \leq 1 + 16I\xi$, so on collapsing the telescoping product we obtain

$$\sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} p(S_1, \ldots, S_I, f) \leq \sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} \frac{1+16I\xi}{e(G)} \prod_{r=2}^{I} \binom{n/2^{r-2} - k}{n/2^{r-1} - k}^{-1}.$$

All terms of this sum are equal, and there are precisely $\prod_{r=0}^{I-2}\binom{n/2^r-k}{n/2^{r+1}-k}$ terms, so

$$\sum_{\substack{S_1\supset\cdots\supset S_I\supset f\\ |S_r|=n/2^{r-1}}} p(S_1,\ldots,S_I,f) \le \frac{1+16I\xi}{e(G)} \le \frac{1+\varepsilon/2}{e(G)}.$$

Hence by (17), we have $\mathbb{P}(F=f)\le(1+\varepsilon)/e(G)$, as required.

Similarly, it follows from (21) that

$$\sum_{\substack{S_1\supset\cdots\supset S_I\supset f\\ |S_r|=n/2^{r-1}}} p(S_1,\ldots,S_I,f) \ge \sum_{\substack{S_1\supset\cdots\supset S_I\supset f\\ |S_r|=n/2^{r-1}}} \frac{1-\delta}{e(G[S_I])}\prod_{r=2}^{I}\frac{(1-6\xi)e(G[S_r])}{\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k}e(G[S_{r-1}])} - \sum_{r=1}^{I}\big(1-\mathbb{P}(\mathcal{E}_r\mid\mathcal{E}_1,\ldots,\mathcal{E}_{r-1})\big).$$

By (16), the last term is bounded above by $\varepsilon/4n^k$; it follows that

$$\sum_{\substack{S_1\supset\cdots\supset S_I\supset f\\ |S_r|=n/2^{r-1}}} p(S_1,\ldots,S_I,f) \ge \sum_{\substack{S_1\supset\cdots\supset S_I\supset f\\ |S_r|=n/2^{r-1}}} \frac{1-6I\xi}{e(G)}\prod_{r=2}^{I}\binom{n/2^{r-2}-k}{n/2^{r-1}-k}^{-1} - \frac{\varepsilon}{4n^k}$$
$$= \frac{1-6I\xi}{e(G)} - \frac{\varepsilon}{4n^k} \ge \frac{1-\varepsilon}{e(G)}.$$

It therefore follows from (17) that $\mathbb{P}(F=f)\ge(1-\varepsilon)/e(G)$ as required. □

It is now easy to prove Theorem 2 from Lemma 24.

Theorem 2 (restated)**.** *There is a randomised algorithm* $\texttt{Sample}(G,\varepsilon)$ *which, given a rational number* ε *with* $0<\varepsilon<1$ *and coloured oracle access to an* n*-vertex* k*-hypergraph* G *containing at least one edge, outputs either a random edge* $f\in E(G)$ *or* `Fail`*. For all* $f\in E(G)$*,* $\texttt{Sample}(G,\varepsilon)$ *outputs* f *with probability* $(1\pm\varepsilon)/e(G)$*; in particular, it outputs* `Fail` *with probability at most* ε*. Moreover, writing* $T=\varepsilon^{-2}k^{7k}\log^{4k+11}n$*,* $\texttt{Sample}(G,\varepsilon)$ *runs in time* $\mathcal{O}(nT)$ *and uses at most* $\mathcal{O}(T)$ *queries to* cIND_G*.*

Proof. To evaluate $\texttt{Sample}(G,\varepsilon)$, we first make n into a power of two by adding at most n isolated vertices to G; note that this does not impede the evaluation of cIND_G. We then call $\texttt{HelperSample}(G,\varepsilon/3)$. If it returns `Fail`, or does not return a value within $\mathcal{O}(nT)$ time and $\mathcal{O}(T)$ oracle queries, then we return `Fail`. Otherwise, we return its output. Writing F for our output, by Lemma 24, for all $f\in E(G)$, we have $\mathbb{P}(F=f)\le(1+\varepsilon)/e(G)$ and

$$\mathbb{P}(F=f)\ge\frac{1-\varepsilon/3}{e(G)}-\frac{2\varepsilon}{3n^k}\ge\frac{1-\varepsilon}{e(G)},$$

as required. □

6. Proofs of Corollaries of Theorems 1 and 2

6.1. **Application to Graph Motif.** In this section we describe how our approach can be used to transform the best known decision algorithm for the Graph Motif problem into an efficient algorithm to approximately count or sample solutions. The Graph Motif problem is stated formally as follows.

Graph Motif
Input: A graph G on n vertices and m edges, equipped with a colouring $c:V(G)\to C$ (where C is some colour-set), and a multiset M consisting of elements of C, with $|M|=k$.

Question: Is there a set $U \subseteq V(G)$ with $|U| = k$ such that U induces a connected subgraph of G and the multiset $\{c(u) : u \in U\}$ is equal to M?

Björklund, Kaski and Kowalik [9] gave the fastest known randomised algorithm to solve (a generalisation of) this problem; in the following theorem, $\mu = \mathcal{O}(\log k \log\log k \log\log\log k)$ accounts for the time required to carry out arithmetic operations in a finite field of size $\mathcal{O}(k)$ and characteristic 2.

Theorem 25. *There exists a Monte Carlo algorithm for* GRAPH MOTIF *that runs in $\mathcal{O}(2^k k^2 m\mu)$ time and in polynomial space, with the following guarantees: (i) the algorithm always returns* `No` *when given a* `No`*-instance as input, (ii) the algorithm returns* `Yes` *with probability at least 1/2 when given a* `Yes`*-instance as input.*

We begin by outlining how we can use their algorithm to obtain a fast decision algorithm for the "multicolour" version of the problem which, to avoid confusion with the colouring already involved in GRAPH MOTIF, we refer to as PARTITIONED GRAPH MOTIF. In an instance of PARTITIONED GRAPH MOTIF, the input graph G is given together with a partition of its vertices into k sets $V_1, \ldots, V_k$, and in order to be valid a solution U must contain exactly one element from each of these sets.

Lemma 26. *There exists a Monte Carlo algorithm for* PARTITIONED GRAPH MOTIF *that runs in $\mathcal{O}(k^{2k-1}m)$ time and in polynomial space, with the following guarantees: (i) the algorithm always returns* `No` *when given a* `No`*-instance as input, (ii) the algorithm returns* `Yes` *with probability at least 1/2 when given a* `Yes`*-instance as input.*

Proof. Write $M = \{c_1, \ldots, c_k\}$. For each possible bijection $\pi : [k] \to [k]$ we will determine, with probability at least $1 - \frac{1}{2k!}$, whether there is a solution $\{v_1, \ldots, v_k\}$ such that for all $i \in [k]$, $v_i \in V_i$ and $c(v_i) = c_{\pi(i)}$. Note that in total we have to consider $k!$ bijections.

For a given bijection π, we achieve this by solving a new instance of GRAPH MOTIF using the algorithm of Theorem 25: we use the same input graph G, but define a new colouring $c' : V(G) \to C \times [k]$, where $c'(v) = (c, i)$ if and only if $c(v) = c$ and $v \in V_i$, and set $M = \{(c_{\pi(1)}, 1), \ldots, (c_{\pi(k)}, k)\}$. To achieve the claimed failure probability it suffices to call the randomised algorithm for GRAPH MOTIF $\lceil k \log k \rceil$ times.

We return `Yes` if any of our trials (over all possibilities for π) returns `Yes`; otherwise we return `No`. By a union bound, the probability that we obtain the correct answer for all choices of π is at least 1/2, and in this case we will output the correct answer.

In total we invoke the randomised GRAPH MOTIF algorithm $k!\lceil k \log k \rceil$ times, so the total running time is $\mathcal{O}(k! k \log k \cdot 2^k k^2 m \log k \log\log k \log\log\log k) = \mathcal{O}(k^{2k-1}m)$. □

This result, combined with Theorem 1, gives us the following corollary; we assume without loss of generality that the input graph is connected and hence that $n = \mathcal{O}(m)$.

Corollary 10 (Restated)**.** *Given an n-vertex instance of* GRAPH MOTIF *with parameter k and $0 < \varepsilon < 1$, there is a randomised algorithm to ε-approximate the number of motif witnesses or to draw an ε-approximate sample from the set of motif witnesses in time $\mathcal{O}(\varepsilon^{-2} k^{8k} m \log^{4k+8} n)$.*

Proof. Let (G, c, M) be an input to #GRAPH MOTIF, and let $0 < \varepsilon < 1$. If $\varepsilon < n^{-k}$ then we can solve either problem by brute force in time $\mathcal{O}(\varepsilon^{-1})$, so suppose not. Let H be the k-hypergraph with vertex set $V(G)$, where $f \subseteq V(G)$ is an edge of H if and only if it is a motif witness for M; thus our desired output is either $e(H)$ or a uniform sample from $E(H)$. We can evaluate $\text{cIND}_H(V_1, \ldots, V_k)$ accurately with probability at least $1 - 1/n^{5k}$ and in time $\mathcal{O}(k^{2k} m \log n)$ by applying the PARTITIONED GRAPH MOTIF algorithm of Lemma 26 to G, c, M and $V_1, \ldots, V_k$ a total of $\lceil 5k \log n \rceil$ times and taking the most common result, breaking ties arbitrarily.

To ε-approximate the number of motif witnesses, we now return the output of $\texttt{Count}(H, \varepsilon, 5/6)$; this returns a valid answer with probability at least $5/6$ unless our simulation of cIND_G fails, which by Theorem 1 happens with probability at most $\varepsilon^{-2}\text{polylog}(n)/n^{3k} \leq 1/6$ for sufficiently large n. For smaller n, we can

solve the problem by brute force, so our overall failure probability is at most $1/3$. Our bound on the running time follows from Theorem 1.

To draw an ε-approximate sample from the set of motif witnesses, we return the output of $\texttt{Sample}(H, \varepsilon/2)$. Writing m for the number of motif witnesses, denote our output by F; then for all motif witnesses f, conditioned on our simulation of cIND_G not failing, we have $F = f$ with probability $(1 \pm \varepsilon/2)/m$. The probability that our simulation fails is at most $\varepsilon^{-2}\mathrm{polylog}(n)/n^{5k} \leq \varepsilon/2n^k \leq \varepsilon/2m$ when n is sufficiently large, so our output is an ε-approximate sample as required. □

6.2. **Application to k-SUM.** In this section, we describe how Theorems 1 and 2 can be used to transform any decision algorithm for k-SUM into an approximate counting or sampling algorithm with roughly the same running time. For all integer $k \geq 3$, the k-SUM problem is stated formally as follows.

> **k-SUM**
> *Input:* A set X of integers.
> *Question:* Is there a set $S \subseteq X$ with $|S| = k$ such that $\sum_{x \in S} x = 0$? We call such sets *witnesses* for X.

We use the following folklore reduction from k-SUM to its multicoloured version.

Corollary 5 (Restated)**.** *Fix $k \geq 3$, suppose an n-integer instance of k-SUM can be solved in time $T(n)$, and write W for the set of witnesses. Then there is a randomised algorithm to ε-approximate $|W|$, or draw an ε-approximate sample from W, in time $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(n))$.*

Proof. We only set out the approximate counting algorithm, since the approximate sampling algorithm is exactly the same (using Theorem 2 in place of Theorem 1). Let us also assume that our decision algorithm for k-SUM is deterministic; otherwise, as in the proof of Corollary 10, we accept an extra factor of $\mathcal{O}(\log n)$ in the running time to drive its failure probability down to $1/n^{5k}$, and then the same algorithm works.

Let X be an instance of k-SUM. Define a k-hypergraph G with vertex set X, whose edges are the sets $S \subseteq X$ with $|S| = k$ such that $\sum_{x \in S} x = 0$. By Theorem 1, there is an algorithm to ε-approximate the number of witnesses for X with probability at least $2/3$ and running time $\mathcal{O}(n\varepsilon^{-2}\,\mathrm{polylog}(n))$ which invokes the coloured independence oracle of G no more than $\mathcal{O}(\varepsilon^{-2}\,\mathrm{polylog}(n))$ times.

To simulate the coloured independence oracle of G, we use our decision algorithm for k-SUM together with a folklore reduction. Let $X_1, \ldots, X_k \subseteq V(G)$ be disjoint. For each $i \in [k]$, define injections $f_i \colon \mathbb{Z} \to \mathbb{Z}$ by

$$f_i(x) = (k+1)^k x + (k+1)^{i-1} \text{ for all } i \in [k-1],$$
$$f_k(x) = (k+1)^k x - \sum_{i=1}^{k-1} (k+1)^{i-1}.$$

For all $i \in [k]$, let $Y_i = \{f_i(x) \colon x \in X_i\}$, and let $Y = Y_1 \cup \cdots \cup Y_k$. Then we evaluate $\mathrm{cIND}_G(X_1, \ldots, X_k)$ by applying our k-SUM decision algorithm to Y and outputting the result; this takes time $\mathcal{O}(n) + T(n)$. We claim edges in $G[X_1, \ldots, X_k]$ correspond to witnesses for Y and vice versa; this implies that the output is correct. Indeed, suppose $\{x_1, \ldots, x_k\}$ is an edge of $G[X_1, \ldots, X_k]$ with $x_i \in X_i$ for all $i \in [k]$. Then $\sum_i x_i = 0$, so

$$\sum_{i=1}^{k} f_i(x_i) = (k+1)^k \sum_{i=1}^{k} x_i + \sum_{i=1}^{k-1} (k+1)^{i-1} - \sum_{i=1}^{k-1} (k+1)^{i-1} = 0,$$

and there is a witness for Y. Conversely, suppose $\{y_1, \ldots, y_k\}$ is a witness for Y. Then by the uniqueness of base-$(k+1)$ expansions, we must have $y_i \in Y_{\sigma(i)}$ for some permutation $\sigma \colon [k] \to [k]$; moreover, $\sum_i f_{\sigma(i)}^{-1}(y_i) = 0$. Thus $\{f_{\sigma(1)}^{-1}(y_1), \ldots, f_{\sigma(k)}^{-1}(y_k)\}$ is an edge of $G[X_1, \ldots, X_k]$ as required. By simulating the coloured independence oracle of G in this way, we obtain an algorithm to ε-approximate the number of witnesses for X with probability at least $2/3$ in running time $\mathcal{O}(n\varepsilon^{-2}\mathrm{polylog}(n) + T(n)\varepsilon^{-2}\mathrm{polylog}(n))$.

Finally, we observe that any algorithm for k-SUM must read a constant proportion of its input, so necessarily $T(n) = \Omega(n)$; thus our running time is $\mathcal{O}(\varepsilon^{-2}T(n)\operatorname{polylog}(n))$. □

6.3. **Application to weighted subgraphs.** Recall that a graph H is a *core* if every homomorphism from H to itself is an automorphism. In this section, we describe how Theorems 1 and 2 can be used to transform any decision algorithm for finding cores with zero edge-weight into an approximate counting or sampling algorithm with roughly the same running time. We state our problem as follows for all graphs H.

EXACT-WEIGHT-H
Input: An edge-weighted graph G with (perhaps negative) integer weights.
Question: Does there exist a subgraph of G isomorphic to H with total weight zero?

Lemma 27. *Let H be a fixed core, suppose that an n-vertex m-edge instance of* EXACT-WEIGHT-H *with weights in $[-M, M]$ can be solved in time $T(m, n, M)$, and write $\mathcal{S}$ for the set of all size-k subsets $S \subseteq V(G)$ which span at least one zero-weight H-subgraph in G. Then there is a randomised algorithm to ε-approximate $|\mathcal{S}|$, or draw an ε-approximate sample from $\mathcal{H}$, in time $\mathcal{O}(\varepsilon^{-2}T(m, n, M)\operatorname{polylog}(n))$.*

Proof. As in Section 6.2, we only set out the approximate counting algorithm, and we suppose for simplicity that our decision algorithm is deterministic. Let G be an instance of EXACT-WEIGHT-H, and let $k = |V(H)|$. Define a k-hypergraph G^+ with vertex set $V(G)$, whose edges are the sets $S \subseteq V(G)$ with $|S| = k$ such that $G[S]$ contains at least one zero-weight H-subgraph. By Theorem 1, since $|V(H)|$ is a constant, there is an algorithm to ε-approximate $e(G^+) = |\mathcal{S}|$ with probability at least $2/3$ and running time $\mathcal{O}(n\varepsilon^{-2}\operatorname{polylog}(n))$ which invokes the coloured independence oracle of G^+ no more than $\mathcal{O}(\varepsilon^{-2}\operatorname{polylog}(n))$ times.

To simulate the coloured independence oracle of G^+, we use our decision algorithm for EXACT-WEIGHT-H together with a simple reduction. Let $X_1, \dots, X_k \subseteq V(G)$ be disjoint, and let $c_G \colon X_1 \cup \dots \cup X_k \to [k]$ be the colouring of $G[X_1 \cup \dots \cup X_k]$ mapping each vertex in X_i to i. For each bijective colouring $c_H \colon V(H) \to [k]$, form a graph $G(c_H)$ from G by removing all vertices outside $X_1 \cup \dots \cup X_k$ and all edges $\{u, v\}$ such that the corresponding edge $\{c_H^{-1}(c_G(u)), c_H^{-1}(c_G(v))\}$ is not present in H. We then apply our decision algorithm to each $G(c_H)$. We claim that $e(G^+[X_1, \dots, X_k]) > 0$ if and only if at least one output is `Yes`. Since $|V(H)|$ is constant, this allows us to evaluate $\mathrm{cInd}_{G^+}(X_1, \dots, X_k)$ in time $\mathcal{O}(m + T(m, n, M))$.

Suppose $e(G^+[X_1, \dots, X_k]) > 0$, so that there exists a multicolour zero-weight H-subgraph K of $G[X_1 \cup \dots \cup X_k]$ embedded by some $\phi \colon V(H) \to V(G)$; then on taking the matching colouring $c_H(v) = c_G(\phi(v))$, we have $K \subseteq G(c_H)$ and so our decision algorithm outputs `Yes` on $G(c_H)$. Conversely, suppose at least one decision algorithm outputs `Yes`; then there exists a bijective colouring $c_H \colon V(H) \to [k]$ such that $G(c_H)$ contains a zero-weight H-subgraph K, embedded by some $\phi \colon V(H) \to V(G(c_H))$. Let $\psi \colon V(G(c_H)) \to V(H)$ map each vertex in X_i to $v_{c_H^{-1}(i)}$, i.e. to the vertex of the same colour in H. Certainly ϕ is a homomorphism, and ψ is a homomorphism by the construction of $G(c_H)$, so $\psi \circ \phi$ is a homomorphism from H to itself; thus, since H is a core, $\psi \circ \phi$ is an automorphism. It follows that ψ is a bijection, so the vertices of K receive different colours under c_G, and K is also a multicolour zero-weight H-subgraph of $G[X_1 \cup \dots \cup X_k]$. It follows that $e(G^+[X_1, \dots, X_k]) > 0$, as required. By simulating the coloured independence oracle of G^+ in this way, we obtain an algorithm to ε-approximate $|\mathcal{S}|$ with probability at least $2/3$ in running time $\mathcal{O}(m\varepsilon^{-2}\operatorname{polylog}(n) + T(m, n, M)\varepsilon^{-2}\operatorname{polylog}(n))$.

Finally, we observe that any algorithm for EXACT-WEIGHT-H must read a constant proportion of its input's edges, so necessarily $T(m, n, M) = \Omega(m)$; thus our running time is $\mathcal{O}(\varepsilon^{-2}T(m, n, M)\operatorname{polylog}(n))$. □

Corollary 6 is immediate from Lemma 27 on taking H to be a k-clique, as each set of k vertices can span at most one zero-weight k-clique. To obtain Corollary 7 from Lemma 27, we apply the EXACT-WEIGHT-H algorithm of [2, Corollary 5] to draw ε-approximate samples S from $\mathcal{S}$ in time $\varepsilon^{-2}\widetilde{\mathcal{O}}(n^{\gamma(H)})$. Given a sample S, we count the number n_S of zero-weight H-subgraphs in $G[S]$ by exhaustive search, then apply

rejection sampling; thus we accept S with probability n_S/k^k, and otherwise we reject S and re-sample it. Thus by choosing our parameters appropriately, we obtain a set $S \in \mathcal{S}$ drawn with probability proportionate to $(1 \pm \varepsilon)n_S$. We then output a uniformly-chosen copy of H from $G[S]$, which is an ε-approximate sample as required, and our expected running time is $\varepsilon^{-2}\widetilde{\mathcal{O}}(k^k n^{\gamma(H)}) = \varepsilon^{-2}\widetilde{\mathcal{O}}(n^{\gamma(H)})$. We can turn this into a deterministic bound on the running time in the usual way, by aborting execution if it runs for too long and applying probability amplification.

6.4. **Application to first-order model checking.** A *vocabulary* is a tuple $\nu = (R_1, \ldots, R_r, \alpha_1, \ldots, \alpha_r)$, where $R_1, \ldots, R_r$ are symbols and $\alpha_1, \ldots, \alpha_r$ are positive integers. A *structure* for ν is a *universe* U together with a set $\mathcal{R}$ of tuples on U. Each tuple receives a non-empty set of labels from $\{R_1, \ldots, R_r\}$, with the property that if a tuple t receives label R_i then $|t| = \alpha_i$. A first-order formula has vocabulary ν if it uses only relations contained in $\{R_1, \ldots, R_r\}$, where for all $i \in [r]$ the relation R_i has arity α_i. For all positive integers k, we define the following problem.

> k-FO PROPERTY TESTING
> *Input:* A vocabulary $\nu = (R_1, \ldots, R_r, \alpha_1, \ldots, \alpha_r)$ with $\alpha_i \leq k$ for all k; a structure $S = (U, \mathcal{R})$ on ν (where $\mathcal{R}$ is given as a list); and a first-order formula ϕ in prenex normal form and with vocabulary ν, free variables $x_1, \ldots, x_\ell$, and quantifier rank at most $k - \ell$.
> *Question:* When each symbol R_i is interpreted as a relation given by the R_i-labelled edges of G, do there exist $y_1, \ldots, y_\ell \in U$ such that assigning $x_i = y_i$ for all $i \in [\ell]$ makes ϕ true?

In this section, we use Theorems 1 and 2 to transform any algorithm for k-FO PROPERTY TESTING into an algorithm for approximately counting or sampling satisfying assignments $(y_1, \ldots, y_\ell)$ with roughly the same running time.

Corollary 3 (Restated)**.** *Fix $k \in \mathbb{Z}_{\geq 0}$, suppose an instance of property testing for k-FO can be solved in time $T(n, m) = \mathcal{O}((m + n)^k)$, where n is the size of the universe and m is the number of tuples in the structure, and write $\mathcal{S}$ for the set of satisfying assignments. Then there is a randomised algorithm to ε-approximate $|\mathcal{S}|$, or draw an ε-approximate sample from $\mathcal{S}$, in time $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(n, m))$.*

Proof. As in Section 6.2, we only set out the approximate counting algorithm, and we suppose for simplicity that our decision algorithm is deterministic. Let (ν, S, ϕ) be an instance of k-FO PROPERTY TESTING, let $S = (U, \mathcal{R})$, and let ℓ be the number of free variables in ϕ. After $\mathcal{O}(m)$ preprocessing time, we may assume that every element of U is contained in some tuple in $\mathcal{R}$ and in particular that $m \geq n/k$.

For all $i \in [\ell]$, let $U_i = U \times \{i\}$. Define an ℓ-hypergraph G with vertex set $U_1 \cup \cdots \cup U_\ell$ whose edges are the sets $\{(y_1, 1), \ldots, (y_\ell, \ell)\}$ such that setting $x_i = y_i$ for all $i \in [\ell]$ makes ϕ true, i.e. such that $(y_1, \ldots, y_\ell)$ is a satisfying assignment of ϕ. By Theorem 1, there is an algorithm to ε-approximate the number of satisfying assignments with probability at least $2/3$ and running time $\mathcal{O}(n\varepsilon^{-2}\,\mathrm{polylog}(n))$ which invokes the coloured independence oracle of G no more than $\mathcal{O}(\varepsilon^{-2}\,\mathrm{polylog}(n))$ times.

To simulate the coloured independence oracle of G, we use our decision algorithm for k-FO PROPERTY TESTING. Let $X_1, \ldots, X_\ell \subseteq V(G)$ be disjoint. Since G is an ℓ-partite ℓ-hypergraph with vertex classes $U_1, \ldots, U_\ell$, the problem of evaluating $\mathrm{cIND}_G(X_1, \ldots, X_\ell)$ reduces to the case where $X_i \subseteq U_i$ for all $i \in [\ell]$. We form a new instance (ν', S', ϕ') of k-FO PROPERTY TESTING as follows: form ν' from ν by adding ℓ additional relation symbols $\in_{X_1}, \ldots, \in_{X_\ell}$, each with arity 1; form S' from S by adding a tuple (y) with label $\in_{X_i}$ for each $(y, i) \in X_i$ (or adding $\in_{X_i}$ to the label set of (y) if $(y) \in \mathcal{R}$ already); and form ϕ' from ϕ by conjoining it with the formula $(\in_{X_1}(x_1) \wedge \cdots \wedge \in_{X_\ell}(x_\ell))$. Then $e(G[X_1, \ldots, X_\ell]) > 0$ if and only if (ν', S', ϕ') is a `Yes` instance, so we can evaluate $\mathrm{cIND}_G(X_1, \ldots, X_\ell)$ in time $\mathcal{O}(n + T(m + n, n))$.

Overall, with preprocessing included, our algorithm runs in time $\mathcal{O}(m + (n + T(m+n, n))\varepsilon^{-2}\,\mathrm{polylog}(n))$. Since $m \geq n/k$ and $T(m, n) = \mathcal{O}((m + n)^k)$, this running time is $\mathcal{O}((m + T(m, n))\varepsilon^{-2}\,\mathrm{polylog}(n))$. Moreover, since any algorithm for k-FO PROPERTY TESTING must read a constant proportion of its input's tuples, we have $T(m, n) = \Omega(m)$; thus the result follows. □

6.5. **Application to k-ORTHOGONAL VECTORS.** In this section, we describe how Theorems 1 and 2 can be used to transform any decision algorithm for k-OV into an approximate counting or sampling algorithm with roughly the same running time. For all integer $k \geq 2$, the k-OV problem is stated formally as follows.

> **k-OV**
> *Input:* k sets $S_1, \ldots, S_k$ of vectors in $\{0,1\}^D$.
> *Question:* Do there exist $x_1 \in X_1, \ldots, x_k \in X_k$ such that $\sum_{j=1}^{D} \prod_{i=1}^{k} (x_i)_j = 0$? We call such tuples $(x_1, \ldots, x_k)$ *witnesses* for $X_1, \ldots, X_k$.

Corollary 4 (Restated)**.** *Fix $k \geq 2$, suppose an N-vector D-dimension instance of k-OV can be solved in time $T(N, D)$, and write W for the set of witnesses. Then there is a randomised algorithm to ε-approximate $|W|$, or draw an ε-approximate sample from W, in time $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(N, D))$.*

Proof. Fix $k \geq 2$. As in Section 6.2, we only set out the approximate counting algorithm, and we suppose for simplicity that our decision algorithm is deterministic. Let $(X_1, \ldots, X_k)$ be an instance of k-OV. Define a k-hypergraph G with vertex set $(X_1 \times \{1\}) \cup \cdots \cup (X_k \times \{k\})$ whose edges are the sets $\{(x_1, 1), \ldots, (x_k, k)\}$ with $\sum_{j=1}^{D} \prod_{i=1}^{k} (x_i)_j = 0$; thus the edges of G correspond to witnesses of $X_1, \ldots, X_k$. By Theorem 1, there is an algorithm to ε-approximate $|W|$ with probability at least $2/3$ and running time $\mathcal{O}(N\varepsilon^{-2} \operatorname{polylog}(N))$ which invokes the coloured independence oracle of G no more than $\mathcal{O}(\varepsilon^{-2} \operatorname{polylog}(N))$ times.

To simulate the coloured independence oracle of G, we use our decision algorithm for k-OV. Since G is a k-partite k-hypergraph with vertex classes $(X_1 \times \{1\}, \ldots, X_k \times \{k\})$, the problem of evaluating $\mathrm{cIND}_G(Y_1, \ldots, Y_k)$ reduces to the case where $Y_i \subseteq X_i \times \{i\}$ for all $i \in [k]$. We form a new instance $(Y_1', \ldots, Y_k')$ of k-OV by setting $Y_i' = \{y \colon (y, i) \in Y_i\}$; then $e(G[Y_1, \ldots, Y_k]) > 0$ if and only if $(Y_1', \ldots, Y_k')$ is a `Yes` instance, so we can evaluate $\mathrm{cIND}_G(Y_1, \ldots, Y_k)$ in time $\mathcal{O}(N + T(N, D))$.

Overall, our algorithm runs in time $\mathcal{O}((N + T(N, D))\varepsilon^{-2} \operatorname{polylog}(n))$. Since any algorithm for k-OV must read a constant proportion of its input vectors, we have $T(N, D) = \Omega(N)$; thus the result follows. □

6.6. **Colourful subgraphs.** In this section, we use Theorems 1 and 2 to transform any decision algorithm for COLOURFUL-H into an approximate counting or sampling algorithm with roughly the same running time. For all graphs H with k vertices, the COLOURFUL-H problem is stated formally as follows.

> **COLOURFUL-H**
> *Input:* A graph G with a vertex-colouring $c : V(G) \to [k]$.
> *Question:* Does G contain a (not necessarily induced) subgraph S such that S is isomorphic to H and for all $i \in [k]$ there is some $v \in V(S)$ with $c(v) = i$?

Corollary 8 (Restated)**.** *Let H be a fixed graph, suppose an n-vertex m-edge instance of* COLOURFUL-H *can be solved in time $T(m, n)$, and write $\mathcal{H}$ for the set of colourful H-subgraphs. Then there is a randomised algorithm to ε-approximate $|\mathcal{H}|$, or draw an ε-approximate sample from $\mathcal{H}$, in time $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(m, n))$.*

Proof. As in Section 6.2, we only set out the approximate counting algorithm, and we suppose for simplicity that our decision algorithm for COLOURFUL-H is deterministic. We slightly depart from previous applications of Theorem 1 in that we construct various different hypergraphs and combine the resulting approximate counts.

For each bijective function $d : V(H) \to [k]$, we let $\mathcal{G}_d$ be a k-hypergraph with vertex set $V(G)$. Let $S \subseteq V(G)$ be a k-set with the property that $c|_S : S \to [k]$ is bijective, that is, S is a colourful k-set. For each such S, we let S be an edge of $\mathcal{G}_d$ if the function $(c|_S^{-1} \circ d) : V(H) \to S$ is an injective homomorphism from H to $G[S]$, i.e. $G[S]$ contains H as a subgraph with vertex colours matching d. We now claim that $\sum_d e(\mathcal{G}_d)$ is equal to the number of injective homomorphisms h from H to G with the property that the

image $h(H)$ is a colourful subgraph of G with respect to the colouring c, i.e. the number of ways H can be embedded in G as a colourful subgraph. To this end, simply note that each such $h : V(H) \to V(G)$ has a unique image $S = h(V(H)) \subseteq V(G)$ and a unique node labelling function $d : V(H) \to [k]$ such that $h = c|_S^{-1} \circ d$ holds.

Because $\sum_d e(\mathcal{G}_d)$ is the number of injective homomorphisms from H to G with a colourful image, it is also equal to $|\mathcal{H}| \cdot \mathrm{Aut}(H)$, where $\mathrm{Aut}(H)$ is the number of automorphisms of H. Since H is fixed, we have access to $\mathrm{Aut}(H)$, and can therefore compute an ε-approximation to $|\mathcal{H}|$ by relying on Theorem 1 to produce an ε-approximation $\hat{e}_d$ to each $e(\mathcal{G}_d)$. In the end, our algorithm outputs $\sum_d \hat{e}_d/\mathrm{Aut}(H)$, which is an ε-approximation to $|\mathcal{H}|$.

It remains to implement and analyse the coloured independence oracle for $\mathcal{G}_d$ for fixed d. Let $X_1, \ldots, X_k \subseteq V(G)$ be disjoint. Since $\mathcal{G}_d$ is k-partite under c, the problem of evaluating $\mathrm{cInd}_{\mathcal{G}_d}(X_1, \ldots, X_k)$ reduces to the case where $X_i \subseteq c^{-1}(i)$ for all $i \in [k]$. Now prepare a graph G' from $G[X_1 \cup \cdots \cup X_k]$ as follows: For each $uv \in E(G')$ with $\{d^{-1}(c(u)), d^{-1}(c(v))\} \notin E(H)$, delete uv from G'; thus we delete all edges between colour classes of G whose corresponding vertices are not joined in H. We query the assumed algorithm for COLOURFUL-H on the input graph G' with the (induced) colouring c. Then we claim $e(\mathcal{G}_d[X_1, \ldots, X_k]) > 0$ if and only if (G', c) is a `Yes` instance, so we can evaluate $\mathrm{cInd}_{\mathcal{G}_d}(X_1, \ldots, X_k)$ in time $\mathcal{O}(n + m + T(n, m))$. Since any algorithm for COLOURFUL-H must read a constant proportion of the input vertices and edges, we have $T(n, m) = \Omega(n + m)$, so we evaluate the oracle in time $\mathcal{O}(T(n, m))$. Our running time bounds therefore follow from Theorem 1.

We now prove correctness. Suppose S is an edge of $\mathcal{G}_d[X_1, \ldots, X_k]$. Then by the definition of $\mathcal{G}_d$, $c|_S^{-1} \circ d$ is an injective homomorphism from H to $G[S]$. By the construction of G', it is also an injective homomorphism from H to $G'[S]$, so G' contains a colourful H-subgraph and so (G', c) is a `Yes` instance. Conversely, suppose (G', c) is a `Yes` instance, so that there is a colourful k-set $S \subseteq V(G')$ such that $G'[S]$ contains a subgraph isomorphic to H. In fact, by the construction of G', $G'[S]$ is itself isomorphic to H under a colour-preserving isomorphism, so the function $c|_S^{-1} \circ d$ is an injective homomorphism from H to $G'[S]$. Thus $S \in E(\mathcal{G}_d[X_1, \ldots, X_k])$, so $e(\mathcal{G}_d[X_1, \ldots, X_k]) > 0$ as required. We have proved the claim. □

REFERENCES

[1] Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. More consequences of falsifying SETH and the orthogonal vectors conjecture. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 253–266. ACM, 2018.

[2] Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-sum conjecture. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2013.

[3] Amir Abboud, Richard Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 218–230. SIAM, 2015.

[4] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, July 1995.

[5] V. Arvind and Venkatesh Raman. Approximation algorithms for some parameterized counting problems. In P. Bose and P. Morin, editors, *ISAAC 2002*, volume 2518 of *LNCS*, pages 453–464. Springer-Verlag Berlin Heidelberg, 2002.

[6] Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 38:1–38:21, 2018.

[7] Nadja Betzler, René van Bevern, Michael Fellows, Christian Komusiewicz, and Rolf Niedermeier. Parameterized algorithmics for finding connected motifs in biological networks. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 8(5):1296–1308, 2011.

[8] Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Triangle estimation using polylogarithmic queries. *CoRR*, abs/1808.00691, 2018.

[9] Andreas Björklund, Petteri Kaski, and Łukasz Kowalik. Constrained multilinear detection and generalized graph motifs. *Algorithmica*, 74(2):947–967, Feb 2016.

[10] Anthony Bonato and Pawel Prałat. The good, the bad, and the great: Homomorphisms and cores of random graphs. *Discrete Mathematics*, 309(18):5535–5539, 2009.

[11] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

[12] Timothy M. Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255. SIAM, 2016.

[13] Yijia Chen, Martin Grohe, and Bingkai Lin. The hardness of embedding grids and walls. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science*, pages 180–192, Cham, 2017. Springer International Publishing.

[14] Holger Dell and John Lapinskas. Fine-grained reductions from approximate counting to decision. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 281–288, 2018.

[15] Holger Dell, Marc Roth, and Philip Wellnitz. Counting answers to existential questions. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, volume 132 of *LIPIcs*, pages 113:1–113:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

[16] Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. Counting h-colorings of partial k-trees. *Theor. Comput. Sci.*, 281(1-2):291–309, 2002.

[17] Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2004.

[18] Martin E. Dyer, Leslie Ann Goldberg, and Mark Jerrum. An approximation trichotomy for boolean #CSP. *J. Comput. Syst. Sci.*, 76(3-4):267–277, 2010.

[19] Michael Fellows, Guillaume Fertin, Danny Hermelin, and Stéphane Vialette. Sharp tractability borderlines for finding connected motifs in vertex-colored graphs. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, pages 340–351, 2007.

[20] Ionuţ Florescu. *Probability and Stochastic Processes*. Wiley-Blackwell, 2014.

[21] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[22] Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, April 2004.

[23] Anka Gajentaan and Mark H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Comput. Geom.*, 45(4):140–152, 2012.

[24] Andreas Galanis, Leslie Ann Goldberg, and Mark Jerrum. A complexity trichotomy for approximately counting list H-colorings. *TOCT*, 9(2):9:1–9:22, 2017.

[25] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2162–2181, 2017.

[26] Sylvain Guillemot and Florian Sikora. Finding and counting vertex-colored subtrees. *Algorithmica*, 65(4):828–844, Apr 2013.

[27] Heng Guo, Chao Liao, Pinyan Lu, and Chihao Zhang. Counting hypergraph colourings in the local lemma regime. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 926–939, 2018.

[28] Heng Guo, Chao Liao, Pinyan Lu, and Chihao Zhang. Zeros of holant problems: locations and algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2262–2278, 2019.

[29] Fereydoun Hormozdiari, Iman Hajirasouliha, Noga Alon, Phuong Dao, and S. Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):i241–i249, 07 2008.

[30] Svante Janson, Tomasz Łuczak, and Andrzej Rucinski. *Random Graphs*. John Wiley & Sons, 2000.

[31] Mark Jerrum and Kitty Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *Journal of Computer and System Sciences*, 81(4):702 – 716, 2015.

[32] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004.

[33] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[34] Daniel M. Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-sum and related problems. *J. ACM*, 66(3):16:1–16:18, 2019.

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

[36] Vincent Lacroix, Cristina G. Fernandes, and Marie-France Sagot. Motif search in graphs: Application to metabolic networks. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(4):360–368, October 2006.

[37] Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010.
[38] Kitty Meeks. The challenges of unbounded treewidth in parameterised subgraph counting problems. *Discrete Applied Mathematics*, 198:170 – 194, 2016.
[39] Kitty Meeks. Randomised enumeration of small witnesses using a decision oracle. *Algorithmica*, 81(2):519–540, Feb 2019.
[40] Moritz Müller. Randomized approximations of parameterized counting problems. In *Parameterized and Exact Computation: Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006. Proceedings*, pages 50–59, 2006.
[41] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075. SIAM, 2010.
[42] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.
[43] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47:85–93, 1986.
[44] R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018.
[45] Ryan Williams. Faster decision of first-order graph properties. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 80:1–80:6, 2014.
[46] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPIcs*, pages 17–29. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.