APPROXIMATELY COUNTING AND SAMPLING SMALL WITNESSES USING A COLOURFUL DECISION ORACLE*

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

Abstract. In this paper, we design efficient algorithms to approximately count the number of edges of a given k-hypergraph, and to sample an approximately uniform random edge. The hypergraph is not given explicitly, and can be accessed only through its *colourful independence oracle*: The colourful independence oracle returns yes or no depending on whether a given subset of the vertices contains an edge that is colourful with respect to a given vertex-colouring. Our results extend and/or strengthen recent results in the graph oracle literature due to Beame et al. (ITCS 2018), Dell and Lapinskas (STOC 2018), and Bhattacharya et al. (ISAAC 2019).

Our results have consequences for approximate counting/sampling: We can turn certain kinds of decision algorithms into approximate counting/sampling algorithms without causing much overhead in the running time. We apply this approximate-counting/sampling-to-decision reduction to key problems in fine-grained complexity (such as k-SUM, k-OV and weighted k-Clique) and parameterised complexity (such as induced subgraphs of size k or weight-k solutions to CSPs).

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 [47]. However, even when the counting version of a problem is hard, it is often easy to approximate well. For example, Jerrum, Sinclair and Vigoda [36] 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 [21, 22, 29, 32, 33]. 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 [48] proved that any polynomial-time algorithm to decide Sat

*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. The collaboration began at Dagstuhl Seminar 17341. An extended abstract [18] was presented at SODA 2020.

†Goethe University Frankfurt, Germany, IT University of Copenhagen and BARC, Copenhagen, Denmark dell@uni-frankfurt.de

‡University of Bristol, Bristol, UK (john.lapinskas@bristol.ac.uk)

§University of Glasgow, Glasgow, UK (Kitty.Meeks@glasgow.ac.uk)

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 $\text{poly}(n, \varepsilon^{-1})$ using an NP-oracle. A similar result holds in the parameterised setting, where Müller [45] proved that a size-n instance of any problem in #W[i] with parameter k can be ε-approximated in time $g(k) \cdot \text{poly}(n, \varepsilon^{-1})$ 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 [17] 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 [17]. 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 $\widetilde{\mathcal{O}}(\varepsilon^{-2})$ overhead. (Here and elsewhere, the $\widetilde{\mathcal{O}}$-notation suppresses a factor of $C \log^C n$ for some constant $C > 0$ depending only on the problem statement.) See Section 1.1 for more details.

The reduction of [17] 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 [17], 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 1.2.)

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 [37], 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 1.1 and 1.2, and discuss our edge-counting reduction framework (which is of independent interest). We describe the applications of our main results to fine-grained complexity and parameterised complexity in Section 1.2.

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

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

For any $(X_1, \dots, X_k) \in \mathcal{C}(G)$, we write $G[X_1, \dots, X_k]$ for the k-partite k-hypergraph on $X_1 \cup \dots \cup X_k$ whose edge set is $\{e \in E(G) \colon |e \cap X_i| = 1 \text{ for all } i \in [k]\}$. We define the *colourful independence oracle*[1] 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 colourful 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 *colourful oracle access* to G, and for legibility we write it to have G as an input. Note that given colourful oracle access to G, it is trivial to simulate the colourful independence oracle of $G[X]$ for any $X \subseteq V(G)$ as in [7]. Our main result is as follows.

THEOREM 1.1. *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 colourful 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 1.1 applies to approximate counting problems, consider the problem #k-CLIQUE of counting the number of size-k cliques in an n-vertex graph H. We take G to be the k-hypergraph on vertex set $V(H)$ whose edges 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 edges 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 1.1 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 1.1 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 1.1 would essentially imply a fine-

[1] Arguably, *colourful edge oracle* would be a more natural name, but we stick to the name established in the literature.

grained reduction from exact counting to decision. This is impossible under SETH in our setting; see [17, Theorem 3] for a more detailed discussion.

We extend Theorem 1.1 to approximately-uniform sampling as follows.

THEOREM 1.2. *There is a randomised algorithm* $\texttt{Sample}(G, \varepsilon)$ *which, given a rational number* ε *with* $0 < \varepsilon < 1$ *and colourful 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 $\widetilde{\mathcal{O}}(\varepsilon^{-1})$ factor; see [37]. Unfortunately, it does not apply to Theorem 1.2, as we do not have a fast way to compute the true distribution of `Sample`'s output.

By the same argument as above, Theorem 1.2 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 1.1 and 1.2 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 below.)

Theorems 1.1 and 1.2 are also of independent interest, generalising known results in the graph oracle literature. Colourful independence oracles are a natural generalisation of the bipartite independent set (BIS) oracles introduced in Beame et al. [7] to a hypergraph setting, and when $k = 2$ the two notions coincide. They were first introduced in Bishnu et al. [12] to solve various decision problems in parameterised complexity. The main result of Beame et al. [7, 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 Theorems 1.1 and 1.2 give a total of $\mathcal{O}(\varepsilon^{-2} \log^{19} n)$ queries used, improving their running time for most values of ε, and extending 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. [10]. (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 in which every edge belongs to at most d triangles, one can ε-approximate the number of triangles in G using at most $\mathcal{O}(\varepsilon^{-12} d^{12} \log^{25} n)$ TIS queries; they have subsequently [11] improved this to $\mathcal{O}(\varepsilon^{-4} d^2 \log^{18} n)$. Our Theorem 1.1 gives an algorithm which requires only $\mathcal{O}(\varepsilon^{-2} \log^{19} n)$ TIS queries, with no dependence on d, and which also generalises to approximately counting k-cliques for all fixed k. Again, Theorem 1.2 extends the result to approximately-uniform sampling.

In an independent work made public shortly after this paper appeared on arXiv, Bhattacharya et al. [9] also prove a version of Theorem 1.1 (but not Theorem 1.2) for arbitrary k. Their generalised d-partite independent set oracles are essentially the same as our independence oracles, and their algorithm makes $\mathcal{O}(\varepsilon^{-4} \log^{5k+5} n)$ queries on an n-vertex k-uniform input hypergraph; for comparison, our algorithm makes $\mathcal{O}(\varepsilon^{-2} \log^{4k+7} n)$ queries. Our algorithm therefore trades a slightly worse dependence

on n for a significantly better dependence on ε. As in [10], the authors are motivated by the theory of graph oracles rather than by applications to specific approximation algorithms.

We note in passing that the main result of [17] doesn't quite fit into this setting, as it also makes unrestricted use of edge existence queries. It resembles a version of Theorem 1.1 restricted to $k = 2$ and with slightly lower overhead in n.

If the colourful independence oracle cIND_G of a k-hypergraph G can be simulated by a deterministic algorithm A_G, then Theorems 1.1 and 1.2 immediately yield algorithms for approximately counting and sampling edges of G: Whenever the algorithms in Theorems 1.1 and 1.2 would query cIND_G, they instead run A_G as a black-box. Suppose instead A_G is a randomised algorithm whose error probability is bounded by a constant, say $1/3$. Then we can use standard probability amplification techniques to make sure that, during any execution of the algorithms in Theorems 1.1 and 1.2, it is very likely that *all* queries to cIND_G are answered correctly. For convenience, we encapsulate this argument in the following corollary. We defer the (simple) proof to Section 6.

Corollary 1.3. *There is a randomised algorithm* `SampleCount` *with the following behaviour. Let G be an arbitrary n-vertex k-hypergraph for some n and k, and let A_G be a randomised implementation of the colourful independence oracle of G with worst-case running time T and error probability at most $1/3$. Let $\varepsilon, \delta > 0$. Then* `SampleCount` $(V(G), k, A_G, \varepsilon, \delta)$ *outputs an ε-approximation of $e(G)$ with error probability at most δ and an ε-approximate sample from $E(G)$ with error probability zero. Moreover, the running time of* `SampleCount` *is at most*

$$\varepsilon^{-2} \log^2(1/\delta)(k \log n)^{\mathcal{O}(k)}(n + T)\,.$$

We remark that the worst-case running time of A_G is measured as the maximum possible running time of executions $A_G(X_1, \dots, X_k)$ over all internal random choices of A_G and inputs $X_1, \dots, X_k$. We also mention an analogous corollary for approximately counting and sampling edges of any k-partite hypergraph $G[X_1, \dots, X_k]$ even if G is not k-partite itself; note that we do not require $X_1 \cup \dots \cup X_k = V(G)$.

In the case where the hypergraph G is k-partite, we can make do with a weaker form of the oracle. Given a k-hypergraph $G = (V, E)$, we define the *uncoloured independence oracle* of G to be the function $\mathrm{IND}_G \colon 2^V \to \{0, 1\}$ such that $\mathrm{IND}_G(X) = 1$ if $G[X]$ contains no edges, and $\mathrm{IND}_G(X) = 0$ otherwise. As we show in Section 6, on a k-partite graph it is not hard to simulate cIND_G given access to IND_G.

Corollary 1.4. *There is a randomised algorithm* `PartitionedSampleCount` *with the following behaviour. Let G be an arbitrary n-vertex k-partite k-hypergraph for some n and k with vertex classes $V_1, \dots, V_k$, and let A_G be a randomised implementation of the uncoloured independence oracle of G with worst-case running time T and error probability at most $1/3$. Let $\varepsilon, \delta > 0$. Then* `PartitionedSampleCount` $(V_1, \dots, V_k, k, A_G, \varepsilon, \delta)$ *outputs an ε-approximation of $e(G)$ with failure probability at most δ and an ε-approximate sample from $E(G)$ with error probability zero. The running time of* `PartitionedSampleCount` *is at most $\varepsilon^{-2} \log^2(1/\delta)(k \log n)^{\mathcal{O}(k)}(n + T)$.*

1.2. Applications. Corollaries 1.3 and 1.4 have algorithmic implications for many well-known problems in fine-grained and parameterised complexity. On a formal level, there is no reason not to simply apply Corollaries 1.3 and 1.4. However, it is instructive to tie these corollaries back to the standard informal notion of the "colourful version" of a problem, which we do in the following pair of definitions.

Recall that a counting problem is a function $\#\Pi\colon \{0,1\}^* \to \mathbb{N}$ and its corresponding decision problem is defined via $\Pi = \{x \in \{0,1\}^* \colon \#\Pi(x) > 0\}$.

Definition 1.5. Π *is a* uniform witness problem *if there is a function from instances $x \in \{0,1\}^*$ to uniform hypergraphs G_x such that:*

(i) $\#\Pi(x) = e(G_x)$*;*

(ii) $V(G_x)$ *and the size of edges in* $E(G_x)$ *can be computed from x in time $\widetilde{\mathcal{O}}(|x|)$;*

(iii) there exists an algorithm which, given x and $S \subseteq V(G_x)$, in time $\widetilde{\mathcal{O}}(|x|)$ prepares an instance $I_x(S)$ of Π such that $G_{I_x(S)} = G_x[S]$ and $|I_x(S)| \in \mathcal{O}(|x|)$.

We say Π *is a* colourful uniform witness problem *if G_x is always k-partite, where k is the edge size of G_x. The set $E(G_x)$ is the set of* witnesses *of the instance x.*

For all the problems we consider, there will only be a single natural choice of hypergraph representation, and so we consider this representation to be a part of the problem statement. For example, k-Clique is a uniform witness problem in which for a given instance $x = (H, k)$, G_x is the hypergraph on $V(H)$ whose edges are the k-cliques of H. It is immediate that Definition 1.5(i) and (ii) are satisfied, and Definition 1.5(iii) is satisfied because "induced subgraphs of G_x correspond to sub-instances of x" — that is, for a given instance $x = (H, k)$ of k-Clique and a given $S \subseteq V(G_x) = V(H)$, $I_x(S)$ is the instance $(H[S], k)$. As a second example, Orthogonal Vectors is a colourful uniform witness problem in which the witnesses are pairs of orthogonal vectors from the given input sets, and again induced subgraphs of G_x correspond to sub-instances of x.

Definition 1.6. *Suppose* Π *is a uniform witness problem.* Colourful-Π *is defined as the problem of, given an instance $x \in \{0,1\}^*$ of Π and a partition of $V(G_x)$ into disjoint sets $S_1, \ldots, S_k$, deciding whether $cIND_{G_x}(S_1, \ldots, S_k) = 0$ holds.*

In other words, in Colourful-Π, the goal is to decide whether $G_x[S_1, \ldots, S_k]$ has at least one edge containing one vertex from each colour class S_i. For example, in Colourful-k-Clique, we are given a graph G, an integer k, and a k-colouring of G, and wish to know whether G contains a k-clique with one vertex of each colour. Observe that if Π is already colourful as in Definition 1.5, then Colourful-Π reduces to $k^{O(k)}$ instances of Π, since any colourful edges must respect the existing vertex classes of G_x and since induced subgraphs of G_x correspond to instances of Π. The colourful version of a problem is sometimes referred to as the *multicolour* version [43].

Given an instance of Π, we write n_x for the number of vertices of G_x, k_x for the edge size of G_x, and W_x for the set of witnesses of x. Using this terminology, we can now restate (a slightly simplified form of) Corollaries 1.3 and 1.4 as follows, where we use part (iii) of Definition 1.5 to ensure that $\mathrm{cIND}_{G_x}(S_1, \ldots, S_k)$ can be computed efficiently even when $S_1 \cup \cdots \cup S_k \subsetneq V(G_x)$. We prove this result in section 6.

Theorem 1.7. *Let Π be a uniform witness problem, and let T be any function from instances of Π to the positive reals. Suppose that given an instance x of Π, there is an algorithm to solve* Colourful-Π *on x with error probability at most $1/3$ in time $T(x)$. Suppose also that any such algorithm has running time $\widetilde{\Omega}(|x|)$. Then there is a randomised algorithm which, given an instance x of Π and $\varepsilon > 0$, with running time*

$$\varepsilon^{-2}(k_x \log n_x)^{\mathcal{O}(k_x)} \cdot \max\big\{T(I_x(S))\colon S \subseteq V(G_x)\big\}, \tag{1.1}$$

outputs an ε-approximation to $\#\Pi(x)$ with probability at least $2/3$ and an ε-approximate sample from W_x with error probability zero.

In a typical application of Theorem 1.7, the maximum in (1.1) will be $\widetilde{\mathcal{O}}(T(x))$. Recall that if Π is already colourful, then Colourful-Π reduces to $k^{O(k)}$ instances of Π, so Theorem 1.7 can be used to reduce from colourful approximate counting to colourful decision as well as from uncoloured approximate counting to colourful decision.

One large family of problems to which this meta-theorem can naturally be applied is that of *self-contained k-witness problems* (see [44]); these are essentially a form of uniform witness problems Π in which for any instance x of Π and all $S \subseteq V(G_x)$, the instance $I_x(S)$ *is* a sub-instance of x in a well-defined sense. Thus, every self-contained k-witness problem is a uniform witness problem, but it seems likely that not every uniform witness problem is a self-contained k-witness problem.

Applications in fine-grained complexity. In the fine-grained setting, k is considered to be a fixed constant, so the running-time bound in Theorem 1.7 can be written as $\widetilde{\mathcal{O}}(\varepsilon^{-2} \cdot T(n,k))$.

In [17], fine-grained reductions from approximate counting to decision were shown for the problems Orthogonal Vectors, 3SUM, and Negative-Weight Triangle. In Section 6, we generalise these reductions to k-Orthogonal Vectors, k-SUM, Zero-Weight k-Clique, and other subgraph isomorphism problems, such as Colourful-H, Exact-Weight k-Clique, and Exact-Weight H. Similarly, we also reduce approximate model counting to model checking with respect to k-variable first-order formulas. In each case, we also have a corresponding result for approximate sampling of witnesses.

We prove these results in Section 6 as a corollary of our main results. To do so, we observe that these problems are natural k-witness problems, and that any decision algorithm for each of these problems can be coaxed in a more or less standard way to also solve the colourful variant of the problem in roughly the same time.

Applications in parameterised complexity. In the parameterised setting, we assume that k is taken as the parameter and are therefore interested in the running time of our algorithms as a function of k. We note that our reduction from approximate counting to decision involves only a "fine-grained FPT overhead": we can rewrite the overhead of $\log^{\mathcal{O}(k)} n$ as an overhead of $k^{2k} n^{o(1)}$ by a standard calculation.[2] Thus, whenever there is an FPT implementation of the colourful independence oracle, Theorem 1.7 gives us an FPTRAS (fixed parameter tractable randomised approximation scheme [6]) for the corresponding (colourful or uncoloured) counting problem. For the formal definition of an FPTRAS, and other standard notions in parameterised (counting) complexity, we refer the reader to [26].

The family of uniform witness problems from Definition 1.5 includes numerous problems in parameterised complexity, including weight-k solutions to CSPs, size-k solutions to database queries, and sets of k vertices in a weighted or unweighted graph or hypergraph which induce a sub(hyper)graph with specific properties. We present three concrete applications of Theorem 1.7.

Our first application of Theorem 1.7 is to the Graph Motif problem, introduced by Lacroix, Fernandes and Sagot [40] in the practical context of metabolic networks. While we defer a detailed discussion to Section 6.1, it will be immediate from the definition that Graph Motif is a uniform witness problem, and that Colourful-Graph Motif reduces to Graph Motif (see Lemma 6.3); hence Theorem 1.7 combined with

[2] Indeed, 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 $\log^{\mathcal{O}(k)} n = \mathcal{O}(k^{2k})$.

the best-known algorithm for Graph Motif yields an improved approximate counting algorithm (see Corollary 6.2).

Our second application of Theorem 1.7 is to induced subgraph counting. Many problems in this area are special cases of Induced Subgraph With Property (Φ), abbreviated to ISWP(Φ). This problem asks whether a given graph G contains a k-vertex induced subgraph with the given property Φ; the problem is then parameterised by k. For example, if Φ is the family of all cliques, then ISWP(Φ) is simply k-Clique. There is also a multi-coloured variant of the problem, MISWP(Φ), in which every vertex in G receives one of k colours, and we seek a k-vertex induced subgraph with property Φ and one vertex of each colour. See [43] for a survey of results on exact and approximate #ISWP(Φ) and #MISWP(Φ), and [27] for a more recent complexity classification of #ISWP(Φ) whenever Φ is a hereditary property. Observe that ISWP is a uniform witness problem, where the witnesses are the k-vertex subsets of $V(G)$ which induce copies of graphs satisfying Φ, and Colourful-ISWP(Φ) is simply MISWP(Φ); hence Theorem 1.7 immediately implies that there is an FPTRAS for #MISWP(Φ) and #ISWP(Φ) whenever there is an FPT decision algorithm for MISWP(Φ), and moreover that the running times of these algorithms are the same up to a sub-polynomial factor in the instance size. This improves on the best previously-known result (due to Meeks [44, Corollaries 4.8 and 4.10]) in two ways: firstly, in [44] the result is proved only for properties Φ that are preserved under adding edges; and secondly, in [44] the FPTRAS is slower than the FPT decision algorithm by a polynomial factor.

Our third application of Theorem 1.7 is to the problems Colourful-H and Weighted-H for a given k-vertex graph H. In Colourful-H, we are given a graph G whose vertices are coloured with k colours, and we wish to decide whether G contains a subgraph copy of H in which every colour is represented. In Weighted-H, we are given an edge-weighted graph G, and we wish to decide whether G contains a subgraph copy of H with total weight zero. Unlike our other two applications, the application of Theorem 1.7 in this setting is not straightforward. The natural way to frame these problems as uniform witness problems would be, as with ISWP, to take the witnesses to be vertex sets which induce a copy of H; however, the number of witnesses would then not in general agree with the number of colourful or zero-weighted copies of H as required by Definition 1.5(i). Nevertheless, we are able to use Theorem 1.7 to give a fine-grained reduction from approximate #Colourful-H to Colourful-H and from approximate #Weighted-H to Weighted-H for all graphs H; see Corollary 6.8 and Corollary 6.10 for details.

One reason this third application is interesting is that it throws an existing research question into a new light. In Uncoloured-H, we are given a graph G, and we wish to decide whether G contains a subgraph copy of H. There are easy and well-known fine-grained reductions from Uncoloured-H to Colourful-H and from approximate #Uncoloured-H to approximate #Colourful-H, via colour coding [5, 4]. For some graphs H, such as cliques, there are also simple reductions in the other direction. However, a full proof of equivalence would imply the long-standing dichotomy conjecture for the parameterised embedding problem (see [16] for recent progress on this conjecture). Corollary 6.8 shows that this question is strongly linked to the question of when there is a fine-grained FPT reduction from Colourful-H to Uncoloured-H, and we hope this will spur further research in the area.

Organisation. In the following section, we set out our notation and quote some standard probabilistic results for future reference. We then prove Theorem 1.1 in Section 3.2, using a weaker approximation algorithm which we set out in Section 4.

We then prove Theorem 1.2 (using Theorem 1.1) 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. For all $S \subseteq V$, we write $d_G(S) = |\{e \in E(G) \colon S \subseteq e\}|$ for the degree of S in G. If $S = \{v_1, \dots, v_{|S|}\}$, then we will sometimes write $d_G(v_1, \dots, v_{|S|}) = d_G(S)$.

For all positive integers t, we write $[t] = \{1, \dots, t\}$. 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 colourful independence oracle of G, and colourful 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 colourful oracle access to G can safely call a subroutine that requires colourful 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 2.1 ([14, Theorem 2.8]). *Let $X_1, \dots, X_m$ be independent real random variables, and suppose there exist $a_1, \dots, a_m, b_1, \dots, b_m \in \mathbb{R}$ 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*

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

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

LEMMA 2.2. *Let $X_1, \dots, X_k$ 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 [14, Corollary 2.11] to both X and $-X$, taking $c = M/3$ and $t = z$, then apply a union bound. ∎

The next lemma collates two standard Chernoff bounds.

LEMMA 2.3 ([34, 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 $t \geq 7\mu$, $\mathbb{P}(X \geq t) \leq e^{-t}$.

The next lemma is a standard algebraic bound.

LEMMA 2.4. *Let $N, k \in \mathbb{Z}_{>0}$. If $N \geq 2k^2$, then $\binom{2N-k}{N-k} / \binom{2N}{N} \geq 2^{-k-1}$.*

Proof. We have

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

∎

We now prove a technical lemma, which should be read as follows. We are given the ability to sample from bounded probability distributions $\mathcal{D}_1, \ldots, \mathcal{D}_q$ 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 2.5 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 $i \in [q]$. This lemma will be important for reducing the running time of our algorithm; see the sketch proof in section 3.1 for more details.

LEMMA 2.5. *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*

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

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 2.5 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 2.5, 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 2.2). 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 2.2,

applied to the variables X and $X_{i,j}/t_i$ with $z = \xi x$, it follows that

$$\begin{aligned}\mathbb{P}(|X - x| \geq \xi x) &\leq 2\exp\left(-\frac{3\xi^2 x^2}{6\sum_i \frac{M_i\mu_i}{t_i} + 2M\xi x}\right)\\ &\leq 2\exp\left(-\frac{3\xi^2 x^2}{2\max\left\{6\sum_i \frac{M_i\mu_i}{t_i},\ 2M\xi x\right\}}\right)\\ &= \max\left\{2\exp\left(-\frac{\xi^2 x^2}{4\sum_i \frac{M_i\mu_i}{t_i}}\right),\ 2\exp\left(-\frac{3\xi x}{4M}\right)\right\}. \qquad (2.1)\end{aligned}$$

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

$$\frac{\xi^2 x^2}{4\sum_i \frac{M_i\mu_i}{t_i}} \geq \log(2/\delta) \geq \ln(2/\delta). \qquad (2.2)$$

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

$$\frac{3\xi x}{4M} \geq \frac{3\log(2/\delta)}{\xi} > \ln(2/\delta). \qquad (2.3)$$

The result therefore follows from (2.1), (2.2) and (2.3). ∎

3. The main algorithm. In this section we prove our main approximate counting result, Theorem 1.1. We will make use of an algorithm with a weaker approximation guarantee, whose properties are stated in Lemma 3.3; we will prove this lemma in Section 4.

3.1. Sketch proof. Let G be the input n-vertex k-hypergraph and let ε be the input error tolerance, so that we wish to find an ε-approximation of $e(G)$. The overall aim of our algorithm is to iteratively construct a list L of random induced sub-hypergraphs $G[X_1], \dots, G[X_t]$, together with associated weights $w_1, \dots, w_t$, such that with high probability:

(A1) $\sum_i w_i \cdot e(G[X_i])$ is an ε-approximation to the total number of edges in G; and
(A2) each sub-hypergraph $G[X_i]$ has few enough edges so that we can efficiently compute $e(G[X_i])$ exactly by brute force using the coloured independence oracle; and
(A3) the list L is short enough so that its length does not significantly affect the running time.

Once a list L satisfying (A1)–(A3) is constructed, we can efficiently approximate $e(G)$ by computing each $e(G[X_i])$ using (A2) and taking the weighted sum $\sum_i w_i e(G[X_i])$.

Constructing the list L. We represent L as a list of pairs (w_i, X_i). Initially L contains the single pair $(1, V(G))$, which naturally satisfies (A1). In order to make progress towards a list satisfying (A2) and (A3), we repeatedly modify it in two ways:

`Refine`: To make the sub-hypergraphs smaller, we replace each hypergraph $G[X]$ in the list L by several random sub-hypergraphs $G[X_i]$ induced by half of the vertices of X. That is, each pair (w, X) is replaced by $(w_1, X_1), \ldots, (w_t, X_t)$ where $|X_i| = |X|/2$ holds for all i. The weights w_i are set to maintain (A1) in expectation. This step gets us closer to achieving (A2) and farther from achieving (A3) while maintaining (A1).

`Reduce`: To make the list L shorter, we discard randomly-selected elements from it and re-weight the remaining elements to maintain (A1) in expectation. This step gets us closer to achieving (A3) while maintaining (A1) and (A2).

We alternate between applying `Refine` and `Reduce` to the list until all of (A1)–(A3) are satisfied, at which point we compute and return $\sum_i w_i e(G[X_i])$.

The overall structure of our algorithm for Theorem 1.1 is very similar to that of Beame et al. [7]. The `Cleanup` step of [7] roughly corresponds to the calculation of the weighted sum at the end, and is similar. However, our implementation of `Refine` and `Reduce` differs in two major ways. Firstly, a key component of both `Refine` and `Reduce` is the ability to compute a coarse estimate of the number of edges in each graph $G[X]$, with multiplicative error $\log^{\Theta(k)} n$, and the technique for this used in [7] does not generalise easily to hypergraphs. We provide such an algorithm as `Coarse` and state its properties as Lemma 3.3, but we defer both the sketch proof and the full proof of this lemma to Section 4. Secondly, by better incorporating these estimates into `Refine` and `Reduce`, we improve the running time's dependence on ε from ε^{-4} to ε^{-2}. In order to expand on these differences, we now sketch our implementation of `Refine` and `Reduce` given `Coarse`.

Sketch of `Reduce`. Suppose we have a long list L of pairs (w, X) satisfying (A1), and we wish to make it shorter by randomly discarding and re-weighting elements while maintaining (A1). (In fact, we start by maintaining something a little stronger than (A1) since the quality of approximation will degrade as the algorithm runs, but we ignore this subtlety in our sketch.) We do this by exploiting a statistical technique called importance sampling, previously applied to the $k = 2$ case by Beame et al. [7]. The idea is to use the coarse estimates of each $e(G[X])$ given by `Coarse` to divide the pairs $(w, X) \in L$ into "bins", with one bin for pairs with $we(G[X]) \in [1, 2)$, one bin for pairs with $we(G[X]) \in [2, 4)$, one bin for pairs with $we(G[X]) \in [4, 8)$, and so on for a total of $\mathcal{O}(k \log n)$ bins. We then choose a suitably large t and sample t pairs from each bin uniformly at random (or take the whole bin if it contains less than t pairs), then re-weight each element to ensure that (A1) is maintained in expectation; we can then prove concentration via e.g. Hoeffding's inequality (Lemma 2.1) to conclude that (A1) is maintained with high probability.

The larger we take t to be in `Reduce`, the fewer pairs we can discard from L; thus to minimise our running time, we should take t to be as small as possible while still being able to apply Hoeffding's inequality, and this is essentially the approach of [7]. The key difference in our approach is that we make use of Lemma 2.5, allowing t to vary between bins. If our estimates from `Coarse` say that pairs (w, X) in some bin account for about half of the quantity $\sum_{(w,X)\in L} we(G[X])$ we wish to preserve, then we will take about half our samples from that bin, and correspondingly fewer from the other bins. This allows our algorithm for `Reduce` to maintain a substantially shorter

list than the algorithm used in [7], and this is one of the two reasons for our improved running time.

Sketch of `Refine`. In [7], `Refine` was implemented by keeping the number of vertices the same but halving the number of edges using random colourings. Our implementation is different and works as follows. Suppose we have a list L of pairs (w, X) satisfying (A1), and we wish to halve the size of all sets X while maintaining (A1). Then for each X and any integer $t \geq 1$, we independently choose t uniformly random subsets $X_1, \dots, X_t \subseteq X$ subject to $|X_i| = |X|/2$ for all i. It is not hard to show using Lemma 2.4 that $\mathbb{E}(e(G[X_i])) \approx e(G)/2^k$ holds for all i. Thus, using Hoeffding's inequality for large enough t, we can show that the total number of edges $\sum_{i=1}^{t} e(G[X_i])$ is concentrated around its mean of roughly $te(G)/2^k$. With high probability, we now have $(2^k/t)\sum_{i=1}^{t} e(G[X_i]) \approx e(G)$, so that replacing (w, X) with $\{(2^k w/t, X_i) \colon i \in [t]\}$ maintains (A1). Similarly to `Reduce`, we use Lemma 2.5 along with our estimates from `Coarse` to tailor the value of t to each pair $(w, X) \in L$ and avoid making the list too long in a single iteration of `Refine`, and this is the second reason for our improved running time.

Organisation. The remainder of this section is organised as follows. We begin by setting out the precise format of our list L in Definition 3.1, which (unlike our sketch) also stores the estimates of edge counts from `Coarse`. We then state the pruning algorithm `Reduce` and prove its correctness in Lemma 3.2. After this, we state the behaviour we expect from our coarse counting subroutine `Coarse` in Lemma 3.3 (deferring the proof to Section 4), state the expansion algorithm `Refine` using `Coarse` as a subroutine, and prove its correctness in Lemma 3.4. We then state `HelperCount`, which is essentially our main algorithm, and prove its correctness in Lemma 3.5. Finally, we prove Theorem 1.1. The purpose of `HelperCount` is to move some uninteresting bookkeeping details out of the main proof for readability; namely, the differences between `HelperCount` and `Count` are that `HelperCount` requires the number of vertices n to be a power of 2, has a constant failure probability (rather than δ), and could (with vanishing probability) run more slowly than expected.

3.2. The main algorithm. 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 3.1. *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, `Reduce`, which we will use to keep the length of L small.

Algorithm `Reduce`$(G, b, y, L, \xi, \delta)$**.**

Input: G is an n-vertex k-hypergraph, where n is a power of 2, to which `Reduce` has (only) colourful 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: `Reduce`$(G, b, y, L, \xi, \delta)$ 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 $1 - \delta$, we have $Z(L') \in (1 \pm \xi)Z(L)$.

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

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

(We will show that 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 an arbitrary order, and return L'.

The algorithm `Reduce` improves significantly on the summation reduction algorithm of [7, Lemma 2.5], which in our notation outputs a list of length

$$\Omega(kb^4 \log(\tfrac{1}{\delta}) \log(nb)/\xi^2)$$

compared to our length of $O(k \log(nb) + b^2 \log(\frac{1}{\delta})/\xi^2)$. We obtain this improvement by defining the number t_i of elements to sample from each "bin" L_i of tuples in creating L'_i to depend on the approximate value $w\hat{e} \approx 2^i$ of tuples in L_i. By contrast, [7, Lemma 2.5] defined a single threshold α and sampled $\min\{\alpha, |L_i|\}$ tuples from each L_i.

LEMMA 3.2. `Reduce`$(G, b, y, L, \xi, \delta)$ *behaves as claimed above, has running time* $\mathcal{O}(|L|k^3 \log(nb/\delta))$*, and does not invoke* cIND_G.

Proof. **Running time.** It is clear that `Reduce`$(G, b, y, L, \xi, \delta)$ does not invoke cIND_G. Recall that we work with the word-RAM model, so we can carry out elementary arithmetic operations on $\mathcal{O}(k \log n)$-bit numbers in $\mathcal{O}(k^2)$ time. Thus step (T1) takes time $\mathcal{O}(k^2 a|L|)$, and step (T2) takes time $\mathcal{O}(k^2 a(\log(1/\delta) + |L|))$. Since $|L'_i| = \min\{|L_i|, t_i\}$, steps (T3) and (T4) take time $\mathcal{O}(k^2 a|L|)$. The required bounds follow.

Correctness. In forming L' from L, `Reduce` only updates the first elements (i.e. the weights) of entries of L; since L is a (G, b, y)-list, and the definition of a (G, b, y)-list does not depend on these weights, L' is also a (G, b, y)-list. We next prove (a). We

have

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

$$= 2a + 1 + \frac{16b^2 \log(2/\delta)}{\xi^2 W} \sum_{|i|\le a} 2^i |L_i|. \tag{3.1}$$

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 (3.1) 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 2.5. Before we can apply it, however, we must set our notation and prove the conditions of the lemma hold. Let

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

thus in step (T3) of `Reduce`, for each $i \notin \mathcal{I}$ we choose $L'_i = L_i$, and for each $i \in \mathcal{I}$ we choose L'_i by sampling t_i elements $(w_{i,j}, S_{i,j}, \hat{e}_{i,j})$ from L_i. For each $i \in \mathcal{I}$ and $j \in [t_i]$, let

$$X_{i,j} := w_{i,j} e(G[S_{i,j}]) |L_i|, \qquad M_i := 2^i b |L_i|,$$

$$\mu_i := \sum_{(w,S,\hat{e})\in L_i} w e(G[S]), \qquad \hat{\mu}_i := \sum_{(w,S,\hat{e})\in L_i} w\hat{e}.$$

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 2.5 that with probability at least $1 - \delta$,

$$\sum_{i\in\mathcal{I}} \sum_{j=1}^{t_i} \frac{X_{i,j}}{t_i} \in (1 \pm \xi/2) \sum_{i\in\mathcal{I}} \mu_i. \tag{3.2}$$

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

Plugging our definitions into (3.2), 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

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

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

$$\sum_{|i|\le a}\sum_{(w,S,\hat{e})\in L'_i} we(G[S]) \in (1\pm\xi/2)\sum_{|i|\le a}\sum_{(w,S,\hat{e})\in L_i} we(G[S]). \tag{3.3}$$

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 (3.3) 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 3.3. *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) colourful 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 $1-\delta$,*

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

Using `Coarse`, we now describe our algorithm for turning our (G,b,y)-list L into a $(G,b,y-1)$-list L' with $Z(L') \approx Z(L)$.

Algorithm `Refine`(G,b,y,L,ξ,δ)**.**

Input: G is an n-vertex k-hypergraph, where n is a power of 2, to which `Refine` has (only) colourful 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: `Refine`(G,b,y,L,ξ,δ) 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

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

and $t_i \leftarrow \left\lceil \dfrac{4b^2 w_i \hat{e}_i \log(4/\delta)}{p\xi^2 W} \right\rceil$ 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

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

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

LEMMA 3.4. $\texttt{Refine}(G, b, y, L, \xi, \delta)$ *behaves as claimed above. Moreover, writing*

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

$\texttt{Refine}(G, b, y, L, \xi, \delta)$ *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

$$\sum_{i=1}^{|L|} t_i \le |L| + \sum_{i=1}^{|L|} \frac{4b^2 w_i \hat{e}_i \log(4/\delta)}{p\xi^2 W} = |L| + \frac{4b^2 \log(4/\delta)}{p\xi^2}. \tag{3.4}$$

Since $2^{y-1} \ge 2k^2$, by a standard binomial coefficient bound (Lemma 2.4), we have $p \ge 2^{-k-1}$. Thus, (3.4) implies

$$\sum_{i=1}^{|L|} t_i \le |L| + \frac{2^{k+3} b^2 \log(4/\delta)}{\xi^2} = \Theta(\lambda). \tag{3.5}$$

By Lemma 3.3, 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 (3.5).

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 2.5. We first explain the purpose of p, as defined in `Refine`. Since L is a (G, b, y)-list, we have $|S_i| = 2^y$ and $|S_{i-1}| = 2^{y-1}$, so there are $\binom{2^y}{2^{y-1}}$ possible choices of each set $S_{i,j}$. Moreover, for any given edge $e \in E(G[S_i])$, there are $\binom{2^y - k}{2^{y-1} - k}$ possible choices of $S_{i,j}$ containing e. It follows that any given edge of $G[S_i]$ is present in S_{i-1} with probability precisely p.

We now set up our notation and show that the relevant assumptions hold in order to apply Lemma 2.5. 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 $X_{i,j} \geq 0$. Moreover, since $(w_i, S_i, \hat{e}_i) \in L$, we have $e(G[S_{i,j}]) \leq e(G[S_i]) \leq \hat{e}_i b$, so $X_{i,j} \leq M_i$. Since p is the probability that any given edge in $G[S_i]$ survives in $G[S_{i,j}]$, we have $\mu_i = \mathbb{E}(X_{i,j})$. The $X_{i,j}$'s are independent, we have $0 \leq \hat{\mu}_i \leq \mu_i b$, 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 2.5 that with probability at least $1 - \delta/2$,

$$\sum_{i=1}^{|L|} \sum_{j=1}^{t_i} \frac{X_{i,j}}{t_i} \in (1 \pm \xi) \sum_{i=1}^{|L|} \mu_i. \tag{3.6}$$

Plugging our definitions in, we see (3.6) implies that $Z(L') \in (1 \pm \xi) Z(L)$. Thus,

$$\mathbb{P}(\mathcal{E}_1) \geq 1 - \delta/2. \tag{3.7}$$

By the correctness of `Coarse` (Lemma 3.3) and a union bound over all $1 \leq i \leq |L|$ and all $j \in [t_i]$, we have $\mathbb{P}(\mathcal{E}_2) \geq 1 - \delta/2$. By a union bound with (3.7), we therefore have $\mathbb{P}(\mathcal{E}_1 \cap \mathcal{E}_2) \geq 1 - \delta$ 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 $b \geq 2(4k \log n)^k$, by the correctness of `Coarse` (Lemma 3.3) we have $\hat{e}/b \leq e(G[S]) \leq \hat{e}b$. Thus, L is a $(G, b, y-1)$-list as required by property (a). We have $|L'| = \sum_i t_i$, so (3.5) 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` only has colourful 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 $n \leq 500$, then return $\sum_{Y \subseteq V(G),\ |Y|=k} (1 - \mathrm{cIND}_G(Y))$.

(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 3.5). Note that this is trivially satisfied at the start of the loop.)

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

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

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

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

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

LEMMA 3.5. *With probability at least* $2/3$, $\texttt{HelperCount}(G, \varepsilon)$ *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 3.3 implies that whenever $\texttt{HelperCount}(G, \varepsilon)$ outputs on step (A1) or (A2), correctness holds, so suppose that this does not occur. For all integers $i \geq 0$, 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) $|L| \leq 33k\log(4nb) + 32b^2\log(2/\delta)/\xi^2$.

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 $(1-\xi)^{2I} \geq 1-2I\xi$, and $(1+\xi)^{2I} \leq e^{2I\xi} \leq 1 + 4I\xi$ (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 3.3). For all $i \in [I]$, let $\mathcal{E}_i$ be the event that `Refine` behaves correctly in the ith iteration of step (A4) and `Reduce` behaves correctly in the ith iteration of step (A5). (If the input restrictions of `Refine` or `Reduce` are violated on the ith iteration, then $\mathcal{E}_i$ occurs automatically.) By correctness of `Refine` and `Reduce` (Lemmas 3.2 and 3.4), 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) of $\pi(i)$, 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 `Refine`

(Lemma 3.4) that L'_i is a $(G, b, \log n - i)$-list with

$$Z(L'_i) \in (1 \pm \xi) Z(L_i), \qquad |L'_i| \le |L_i| + 2^{k+3} b^2 \log(4/\delta)/\xi^2. \tag{3.8}$$

We next show that $1/2 \le Z(L'_i) \le 2n^k$, that $|L'_i| \le n^{11k}$, and that $\xi \ge n^{-2k}$, as required by `Reduce`. Since property (ii) of $\pi(i)$ 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 $\texttt{HelperCount}(G, \varepsilon)$ did not halt at (A2), we have $1 \le e(G) \le n^k$; since $\varepsilon < 1/2$, it follows that $1/2 \le Z(L'_i) \le 2n^k$. Since $\texttt{HelperCount}(G, \varepsilon)$ did not halt at (A1), we have $n \ge 500$ and hence $n \ge 50 \log n$. We also have $\varepsilon \ge n^{-k}$ and $k \le n$. Hence:

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

Since property (iii) of $\pi(i)$ holds for L_i, it follows that

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

We have therefore shown that L'_i and ξ satisfy the input restrictions of `Reduce`. Since $\mathcal{E}_i$ occurs, by the correctness of `Reduce` (Lemma 3.2) and by (3.8) it follows that L_{i+1} is a $(G, b, \log n - i)$-list with

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

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 $n \le 500$, 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 $\varepsilon \ge n^{-k}$. 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 $\Lambda = 33k \log(4nb) + 32 b^2 \log(2/\delta)/\xi^2$ for the upper bound on $|L|$ in property (iii) of our invariant π.

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

For step (A1), we use the fact that $\varepsilon \ge n^{-k}$ and so the conditional does not trigger; we verify this fact in time $\mathcal{O}(k^2)$. 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 3.3). For step (A5), we first observe that the step is executed I times. We then apply the time bounds on `Reduce` (Lemma 3.2), together with property (iii) of $\pi(i)$ and the fact that `Refine` adds at most $2^{k+3} b^2 \xi^{-2} \log(4/\delta)$ to the length of L (see Lemma 3.4). (Note

that $k^3 \log(nb/\delta) = \mathcal{O}(k^4 \log n)$.) 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 `Refine` (Lemma 3.4), 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 recall Theorem 1.1 and then prove it.

THEOREM 1.1. *There is a randomised algorithm* `Count`(G, ε, δ) *with the following behaviour. Suppose G is an n-vertex k-hypergraph, and that* `Count` *has colourful 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, `Count`(G, ε, δ) *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 `Count`(G, ε, δ), 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 `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 `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 2.3(i)) we have

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

as required. ∎

4. Coarse approximate counting. In this section, we prove Lemma 3.3. We fix the input graph G to be an n-vertex k-hypergraph to which we have (only) colourful 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 colourful 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 colourful 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. [7].

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 $X_1, \dots, X_k$ independently with each iteration of step (C2).

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

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

Behaviour: Let $p_{\mathsf{out}} = (8k \log n)^{-k}$.
Completeness: If $e(G[X_1, \dots, X_k]) \geq M$, then `VerifyGuess` outputs `Yes` with probability at least p_{out}.
Soundness: If $e(G[X_1, \dots, X_k]) < M \cdot p_{\mathsf{out}}/2(k \log n)^k$, 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 $(a_1, \dots, a_k)$ with $0 \leq a_1, \dots, a_k \leq k \log n$ and $a_1 + \cdots + a_k \geq \log M$.
(C2) For each tuple $(a_1, \dots, a_k) \in A$: If $\mathrm{cIND}_G(Y_{1,a_1}, \dots, Y_{k,a_k}) = 0$, 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 4.1. `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 $G, M, X_1, \dots, X_k$ be the input for `VerifyGuess`, and write

$$H = G[X_1, \dots, X_k]\,.$$

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 $\mathcal{O}(nk^2 \log n + k^k \log^k n)$ 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 $(a_1, \dots, a_k) \in A$, and let $H' \subseteq H$ denote the random induced subgraph $G[Y_{1,a_1}, \dots, Y_{k,a_k}]$. 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 $(a_1, \dots, a_k) \in A$, 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 show 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, recall from Section 2.1 that for all $i \in [k]$ and $v_1, \ldots, v_i \in V(G)$, $d_H(v_1, \ldots, v_i)$ is the number of edges in H containing $\{v_1, \ldots, v_i\}$ as a subset. 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 4.1 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. The lemma statement therefore follows as well.

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

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

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 $Y_{1,a_1}, \dots, Y_{i-1,a_{i-1}}$: Let $\mathcal{F}$ be a possible filtration of these variables consistent with $\mathcal{E}_{i-1}$, and let $a_1, \dots, a_{i-1}$ and $v_1, \dots, v_{i-1}$ 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

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

Every edge in $H[v_1, \dots, v_{i-1}, X_i, \dots, X_k]$ 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

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

edges of $H[v_1, \dots, v_{i-1}, X_i, \dots, X_k]$. We take $a_i := \lceil \log d_H(v_1, \dots, v_{i-1}) \rceil - D_i + 1$; 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 $d_H(v_1, \dots, v_{i-1}, v) \leq 2^{D_i}$, we have $2^{D_i}|X_i^{D_i}| \geq d_H(v_1, \dots, v_{i-1})/k \log n$. It follows that

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

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 3.3. 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 colourful 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 4.2. `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 $N = \mathcal{O}((8k \log n)^{k+1})$. The algorithm `ColourCoarse` executes `VerifyGuess` at most $\mathcal{O}(\log(n^k) N)$ times. By Lemma 4.1, 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 2.3(i) taking $\varepsilon = 1/4$),

$$\mathbb{P}\big(|S_M - \mathbb{E}(S_M)| \geq \mathbb{E}(S_M)/4\big) \leq 2 \exp\big(-\tfrac{1}{48}\mathbb{E}(S_M)\big) \tag{4.1}$$

If $e(H) \geq M$, then the completeness of `VerifyGuess` implies $\mathbb{E}(S_M) \geq N p_{\mathsf{out}}$. Thus, (4.1) and our choice of N imply that

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

Similarly, if $e(H) < M p_{\mathsf{out}}/(2(k \log n)^k)$, then the soundness of `VerifyGuess` implies $\mathbb{E}(S_M) \leq N p_{\mathsf{out}}/2$. But then (4.1) implies that

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

Finally, we perform a union bound over all $M \in \{1, 2, 4, 8, \dots, n^k\}$ that satisfy either $e(H) \geq M$ or $e(H) \leq M p_{\mathsf{out}}/(2k^k \log^k n)$. (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 $S_M > 3N p_{\mathsf{out}}/4$ whenever $e(H) \geq M$ and $S_M < 3N p_{\mathsf{out}}/4$ whenever $e(H) \leq M p_{\mathsf{out}}/(2k^k \log^k n)$. 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 colourful oracle access (only).

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

(E1) Let $t = 12e^{2k}$, 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 $X_1,\dots,X_k$ of $V(G)$.
  (E4) Execute `ColourCoarse`$(G, X_1,\dots,X_k)$ 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 4.3. `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 4.2 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 $G_i = G[c_i^{-1}(1),\dots,c_i^{-1}(k)]$ 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 $x_{i,j}/b \le m_i \le x_{i,j}b$.

Note that the $\mathcal{E}_{i,j}$'s are independent conditioned on c_i, and that the correctness of `ColourCoarse` (Lemma 4.2) implies that $\mathbb{P}(\mathcal{E}_{i,j}) \ge 2/3$ 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 2.3(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 2.1) that

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

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

$$\mathbb{P}\Big(\Big|\frac{k^k}{tk!}\sum_i m_i - e(G)\Big| > \frac{1}{2}e(G)\Big) \leq 2\exp\left(-te^{-2k}/4\right) \leq 1/6.$$

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

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

We now restate Lemma 3.3 and prove it via the usual probability amplification argument.

LEMMA 3.3. *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) colourful 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* $1-\delta$*,*

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

Proof. Given G and $\delta > 0$, we simply invoke $\texttt{HelperCoarse}(G)$ a total of $T := \lceil 36\ln(2/\delta)\rceil$ times and return the median output. By the correctness of `HelperCoarse` (Lemma 4.3), 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 2.3(i) with $\varepsilon = 1/2$ and $\mu = T/3$) that we output a valid approximation with probability at least $1 - 2e^{-T/36} \geq 1-\delta$, as required, and our bounds on running time and oracle usage are immediate from Lemma 4.3. ∎

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 1.2. We begin by sketching our algorithm.

Suppose for the moment that we are given access to a *deterministic* algorithm $\texttt{ExactCount}(G)$ which, given colourful oracle access to a k-hypergraph G, returns the *exact* value of $|E(G)|$. Let G be an n-vertex k-hypergraph for which we have access to a colourful independence oracle, and suppose further that n is a power of two and that G contains at least one edge. In this hypothetical scenario, we could proceed to sample an edge of G from the *exact* uniform distribution using iterated rejection sampling as follows.

Let $X_1 = V(G)$ and $M_1 = \texttt{ExactCount}(G)$. Choose a uniformly random subset $X \subset X_1$ of size $n/2$, and let $M = \texttt{ExactCount}(G[X])$. With probability M/M_1, we "accept" X, setting $X_2 = X$ and $M_2 = M$. Otherwise, we "reject" X, reinitialising it to a new uniformly random size-$(n/2)$ subset, and repeat the process until we accept some set X. Likewise, we then choose a uniformly-random subset $X \subset X_2$ of size $n/4$ and let $M = \texttt{ExactCount}(G[X])$; with probability M/M_2, we "accept" X, setting $X_3 = X$ and $M_3 = M$, and otherwise we "reject" X and resample. Continuing in this way, we generate two sequences $X_1,\dots,X_I$ and $M_1,\dots,M_I$ with $|X_i| = n/2^{i-1}$ for all $i \leq I$, where I is chosen so that $k \leq |X_I| \leq 2k$. We then enumerate the edges of $G[X_I]$ directly using $O(2^k)$ calls to the colourful independence oracle, and output a uniformly random such edge F.

Observe that by a standard rejection sampling argument (see e.g. Florescu [25, Proposition 3.3]), for any possible value $(Y_1, \dots, Y_I, e)$ of $(X_1, \dots, X_I, F)$, we have

$$\mathbb{P}\big((X_1, \dots, X_I, F) = (Y_1, \dots, Y_I, e)\big) = \frac{1}{|E(G[Y_I])|} \cdot \prod_{i=1}^{I-1} \frac{|E(G[Y_{i+1}])|}{\sum_{\substack{Z \subseteq Y_i \\ |Z| = n/2^i}} |E(G[Z])|}.$$

Since for each i, each edge of $G[Y_i]$ contributes 1 to exactly $\binom{|Y_i|-k}{|Y_i|/2-k}$ terms in the sum above and 0 to the rest, it follows that

$$\begin{aligned}\mathbb{P}\big((X_1, \dots, X_I, F) = (Y_1, \dots, Y_I, e)\big) &= \frac{1}{|E(G[Y_I])|} \cdot \prod_{i=1}^{I-1} \frac{|E(G[Y_{i+1}])|}{|E(G[Y_i])|} \binom{|Y_i| - k}{|Y_i|/2 - k}^{-1} \\ &= \frac{1}{|E(G)|} \prod_{i=1}^{I-1} \binom{|Y_i| - k}{|Y_i|/2 - k}^{-1}.\end{aligned}$$

On summing over all possible sequences $Y_1, \dots, Y_I$, we recover that $\mathbb{P}(F = e) = 1/|E(G)|$ as required. Moreover, each edge of $G[X_i]$ appears in $G[X_{i+1}]$ with probability roughly $1/2^k$, so in expectation we will reject roughly 2^k samples before accepting each X_i, for a total of $\mathcal{O}(2^k \log n)$ calls to `ExactCount`.

Of course, `Count` is neither a deterministic nor an exact counting algorithm. However, with careful bookkeeping, essentially the same algorithm turns out to yield an approximate sample. We do the bulk of the work with the following ancillary algorithm `HelperSample`(G, ε), which assumes that $|V(G)|$ is a power of two and that G contains at least one edge; we then address these assumptions (which will be easy) in the main proof of Theorem 1.2 later in the section.

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 colourful 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 $I \le 1$, enumerate the edges of $e(G)$ using cIND_G and return a uniformly random sample.

(S3) Let $X_1 \leftarrow V(G)$, $M_1 \leftarrow$ `Count`(X_1, ξ, δ), and $i \leftarrow 2$. While $i \le I$:
   (a) Choose a size-$(|X_{i-1}|/2)$ set $X \subseteq X_{i-1}$ uniformly at random, and let $M \leftarrow$ `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 5.1. `HelperSample`(G, ε) *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 $\varepsilon \le n^{-k}$ or $I \le 1$, then both correctness and the stated time bounds are clear, so suppose $\varepsilon > n^{-k}$ and $I \ge 2$ (which implies $n \ge 32k^2$). We first carefully bound the probability that something goes wrong over the course of the algorithm's execution.

We define events as follows for all $r \in [I]$:

- Let $\mathcal{E}_r$ be the event that in step (S3) `Count` is called at most $2^{k+3}\ln(\frac{8In^k}{\varepsilon})$ 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 $M_r > 0$.
- Let $\mathcal{E}_{r,1}$ be the event that when $i = r$, we accept a set X_r within $2^{k+3}\ln(\frac{8In^k}{\varepsilon})$ 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 $M \notin (1 \pm \xi)e(G[X])$.

Intuitively, $\mathcal{E}_r$ is the event that the algorithm behaves as we expect in determining X_r. We will use $\mathcal{E}_{r,1}$ and $\mathcal{E}_{r,2}$ to bound $\mathbb{P}(\mathcal{E}_r \mid \mathcal{E}_1, \dots, \mathcal{E}_{r-1})$ below for all $r \in [I]$, and hence bound $\mathbb{P}(\mathcal{E}_1, \dots, \mathcal{E}_r)$ below. When $r = 1$, it follows from Theorem 1.1 and the fact that $e(G) > 0$ that $\mathbb{P}(\mathcal{E}_1) \ge 1 - \delta$.

Let $2 \le r \le I$, let $\mathcal{F}$ be a possible filtration of $X_1, \dots, X_{r-1}$ and $M_1, \dots, M_{r-1}$ compatible with $\mathcal{E}_1, \dots, \mathcal{E}_{r-1}$, and let Y be the value of X_{r-1} determined by $\mathcal{F}$. 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

$$\mathbb{P}(\mathcal{E}_r \mid \mathcal{F}) \ge \mathbb{P}(\mathcal{E}_{r,1} \cap \mathcal{E}_{r,2} \mid \mathcal{F}). \tag{5.1}$$

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 $i = r$. In this iteration, 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; note that each of these events is independent of past samples, and that their probability does not depend on the value of i. By Theorem 1.1, we accept X in any given round with probability at least $\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F}) - \delta$; hence

$$\mathbb{P}(\mathcal{E}_{r,1} \mid \mathcal{F}) \ge 1 - \big(1 - \mathbb{P}(\mathcal{A}_2 \mid \mathcal{F}) + \delta\big)^{2^{k+3}\ln(8In^k/\varepsilon)}. \tag{5.2}$$

Let T be the number of iterations of (S3a)–(S3c) before either `Count` returns an inaccurate estimate or we accept an accurate estimate. By Bayes' theorem, for all $t \ge 0$, we have

$$\begin{aligned}\mathbb{P}(\mathcal{E}_{r,2} \mid \mathcal{F} \text{ and } T = t) &= \frac{\mathbb{P}(\mathcal{E}_{r,2} \text{ and } T = t \mid \mathcal{F})}{\mathbb{P}(T = t \mid \mathcal{F})} = \frac{\mathbb{P}(\mathcal{E}_{r,2} \text{ and } T = t \mid \mathcal{F} \text{ and } T \ge t)}{\mathbb{P}(T = t \mid \mathcal{F} \text{ and } T \ge t)} \\ &= \frac{\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F})}{\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F}) + \mathbb{P}(\mathcal{A}_1 \mid \mathcal{F})}.\end{aligned}$$

Summing over all values of t yields

$$\mathbb{P}(\mathcal{E}_{r,2} \mid \mathcal{F}) = \frac{\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F})}{\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F}) + \mathbb{P}(\mathcal{A}_1 \mid \mathcal{F})}. \tag{5.3}$$

We now bound $\mathbb{P}(\mathcal{A}_1 \mid \mathcal{F})$ above and $\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F})$ below. Theorem 1.1 implies that

$$\mathbb{P}(\mathcal{A}_1 \mid \mathcal{F}) \le \delta. \tag{5.4}$$

Moreover, for all $S \subset Y$ with $|S| = |Y|/2$, Theorem 1.1 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 2.4 it follows that $\mathbb{P}(\mathcal{A}_2 \mid \mathcal{F}) \geq 2^{-k-2}$. It therefore follows from (5.2), (5.3) and (5.4) that

$$\begin{aligned}
\mathbb{P}(\mathcal{E}_{r,1} \mid \mathcal{F}) &\geq 1 - (1 - 2^{-k-2} + \delta)^{2^{k+3}\ln(8In^k/\varepsilon)} \geq 1 - (1-2^{-k-3})^{2^{k+3}\ln(8In^k/\varepsilon)} \\
&\geq 1 - e^{-\ln(8In^k/\varepsilon)} = 1 - \varepsilon/8In^k, \\
\mathbb{P}(\mathcal{E}_{r,2} \mid \mathcal{F}) &\geq 2^{-k-2}/(2^{-k-2}+\delta) \geq 1 - 2^{k+2}\delta \geq 1 - \varepsilon/8In^k.
\end{aligned}$$

By (5.1), it follows that $\mathbb{P}(\mathcal{E}_r \mid \mathcal{F}) \geq 1 - \varepsilon/4In^k$ for all $r \in [I]$. Thus

$$\mathbb{P}(\mathcal{E}_r \mid \mathcal{E}_1, \dots, \mathcal{E}_{r-1}) \geq 1 - \varepsilon/4In^k \text{ for all } r \in [I], \text{ and} \tag{5.5}$$

$$\mathbb{P}(\mathcal{E}_1, \dots, \mathcal{E}_I) \geq 1 - \varepsilon/4n^k. \tag{5.6}$$

With these equations 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 1.1, 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. As by (5.6), with probability at least $1 - \varepsilon/n^k$ there are at most $2^{k+3}\ln(8In^k/\varepsilon) = \mathcal{O}(k2^k\log n)$ such invocations, the claimed bounds follow.

Correctness. Let F be the output of `HelperSample`(G,ε), or `Fail` if it does not halt. Since $e(G) > 0$, by (5.6), `HelperSample`(G,ε) outputs a sample from $E(G)$ with probability at least $1 - \varepsilon/n^k$; thus to prove Lemma 5.1, 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

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

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 (5.6), it follows that

$$\sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} p(S_1, \ldots, S_I, f) \leq \mathbb{P}(F = f) \leq \sum_{\substack{S_1 \supset \cdots \supset S_I \supset f \\ |S_r| = n/2^{r-1}}} p(S_1, \ldots, S_I, f) + \frac{\varepsilon}{4n^k}. \tag{5.7}$$

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 1.1 and the definition of $\mathcal{F}_r$,

$$q(S) \geq (1-\delta)\frac{(1-\xi)e(G[S])}{(1+\xi)e(G[S_{r-1}])} \text{ and } q(S) \leq \frac{(1+\xi)e(G[S])}{(1-\xi)e(G[S_{r-1}])} + \delta. \tag{5.8}$$

Moreover, by a standard rejection sampling argument (see e.g. Florescu [25, Proposition 3.3]), we have

$$\mathbb{P}(X_r = S_r \mid \mathcal{F}_r) = \frac{q(S_r)}{\sum_{\substack{T \subset S_{r-1} \\ |T| = |S_{r-1}|/2}} q(T)}. \tag{5.9}$$

By (5.8) and (5.9), and using the fact that $e(G[S_r]) \geq 1$, we have

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

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

$$\begin{aligned}
\mathbb{P}(X_r = S_r \mid \mathcal{F}_r) &\leq \frac{(1+\xi)^3}{(1-\xi)^3} \cdot \frac{e(G[S_r])}{\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k} e(G[S_{r-1}])} \\
&\leq (1+8\xi)\frac{e(G[S_r])}{\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k} e(G[S_{r-1}])}.
\end{aligned} \tag{5.10}$$

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

Also by (5.8) and (5.9), we have

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

Since there are $\binom{|S_{r-1}|}{|S_{r-1}|/2}$ terms in the sum, and since each edge of $G[S_{r-1}]$ contributes 1 to $\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k}$ terms of the sum and zero to the rest, it follows that

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

Since $|S_{r-1}| = n/2^{r-1} \geq 4k^2$, by Lemma 2.4 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

$$\mathbb{P}(X_r = S_r \mid \mathcal{F}_r) \geq \frac{(1-\xi)^3}{(1+\xi)^2} \cdot \frac{e(G[S_r])}{(1+2^{k+1}\delta)\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k} e(G[S_{r-1}])}$$
$$\geq (1-6\xi) \frac{e(G[S_r])}{\binom{|S_{r-1}|-k}{|S_{r-1}|/2-k} e(G[S_{r-1}])}. \tag{5.11}$$

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 $p(S_1, \dots, S_I, f)$. Observe that for all $f \in E(G)$,

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

It therefore follows from (5.10) that

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

(Recall that $e(G[S_r]) \geq 1$ for all $r \in [I]$.) We have $(1+8\xi)^I \leq e^{8I\xi} \leq 1 + 16I\xi$, so on collapsing the telescoping product we obtain

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

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

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

Hence by (5.7), we have $\mathbb{P}(F=f)\le(1+\varepsilon)/e(G)$, as required.

Similarly, it follows from (5.11) that

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

By (5.5), the last term is bounded above by $\varepsilon/4n^k$; it follows that

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

It therefore follows from (5.7) that $\mathbb{P}(F=f)\ge(1-\varepsilon)/e(G)$ as required. □

It is now easy to prove Theorem 1.2 from Lemma 5.1.

THEOREM 1.2. *There is a randomised algorithm* $\texttt{Sample}(G,\varepsilon)$ *which, given a rational number* ε *with* $0<\varepsilon<1$ *and colourful 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 5.1, 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. Corollaries of Theorems 1.1 and 1.2. In this section, we restate and prove some fairly straightforward consequences of our main results, which allow us to turn decision algorithms for various problems in parameterised and fine-grained complexity into approximate counting and sampling algorithms.

COROLLARY 1.3. *There is a randomised algorithm* `SampleCount` *with the following behaviour. Let* G *be an arbitrary* n*-vertex* k*-hypergraph for some* n *and* k*, and let*

A_G be a randomised implementation of the colourful independence oracle of G with worst-case running time T and error probability at most $1/3$. Let $\varepsilon, \delta > 0$. Then `SampleCount` *$(V(G), k, A_G, \varepsilon, \delta)$ outputs an ε-approximation of $e(G)$ with error probability at most δ and an ε-approximate sample from $E(G)$ with error probability zero. Moreover, the running time of* `SampleCount` *is at most*

$$\varepsilon^{-2} \log^2(1/\delta)(k \log n)^{\mathcal{O}(k)}(n + T)\,.$$

Proof. We may assume without loss of generality that $\varepsilon > n^{-k}$, since otherwise we can count or sample by brute force in time $\mathcal{O}(\varepsilon^{-1}T)$ by applying the (simulated) colourful independence oracle to each k-element subset of $V(G)$.

We next reduce the failure probability of A_G with a standard probability amplification technique. Let A'_G be the algorithm with runs A_G on the given inputs $\lceil 600k \log(n/\delta) \rceil$ times and returns the most common answer. By Lemma 2.3, the probability that this answer is incorrect is at most $2e^{-600k \log(n/\delta)/36} < (n/\delta)^{-14k}$, and each invocation of A'_G takes $\mathcal{O}(Tk \log n)$ time. We now simply run `Count`$(G, \varepsilon, \delta/2)$ and `Sample`$(G, \varepsilon/2)$ and return the results, using A'_G to simulate each call to the colourful independence oracle of G.

Recall the properties of `Count` and `Sample` from Theorems 1.1 and 1.2. In both algorithms, A'_G is invoked $\mathcal{O}(\varepsilon^{-2}k^{7k} \log^{4k+11} n)$ times; by a union bound, for sufficiently large n, the probability it returns the correct answer on every invocation is at least

$$1 - \mathcal{O}\big(\varepsilon^{-2}k^{7k} \log(1/\delta) \log^{4k+11} n \cdot (n/\delta)^{-14k}\big) \geq 1 - (n/\delta)^{-10k}.$$

Conditioned on all oracle calls returning the correct answer, `Count` returns a valid ε-approximation with probability at least $1 - \delta/2$ and `Sample` returns a valid $(\varepsilon/2)$-approximate sample. It follows by union bounds that without this conditioning, `Count` returns a valid ε-approximation with probability at least $1 - \delta/2 - n^{-10k}$, and `Sample` returns a valid $(n^{-10k} + \varepsilon/2)$-approximate sample with no possibility of failure. Since $(n/\delta)^{-10k} \leq \delta/2$ and $(n/\delta)^{-10k} \leq \varepsilon/2$, it follows that `SampleCount` behaves as desired. Moreover, by our running time bounds on `Count`, `Sample` and A'_G, the running time of `SampleCount` is $\varepsilon^{-2}(k \log n)^{\mathcal{O}(k)}(n + T)$ as required. ∎

We believe that Corollary 1.4 does follow from a careful reading of the proofs of Theorems 1.1 and 1.2 in the way one might expect, replacing all randomly-applied colourings of the graph by the colouring induced by $X_1, \dots, X_k$. However, for the benefit of the reader, we give a slightly less efficient but far more easily-checkable algorithm by means of a standard colour coding argument.

Corollary 1.4. *There is a randomised algorithm* `PartitionedSampleCount` *with the following behaviour. Let G be an arbitrary n-vertex k-partite k-hypergraph for some n and k with vertex classes $V_1, \dots, V_k$, and let A_G be a randomised implementation of the uncoloured independence oracle of G with worst-case running time T and error probability at most $1/3$. Let $\varepsilon, \delta > 0$. Then* `PartitionedSampleCount` *$(V_1, \dots, V_k, k, A_G, \varepsilon, \delta)$ outputs an ε-approximation of $e(G)$ with failure probability at most δ and an ε-approximate sample from $E(G)$ with error probability zero. The running time of* `PartitionedSampleCount` *is at most $\varepsilon^{-2} \log^2(1/\delta)(k \log n)^{\mathcal{O}(k)}(n + T)$.*

Proof. By Corollary 1.3, it suffices to simulate the colourful independence oracle of G in time $(k \log n)^{\mathcal{O}(k)}(n+T)$. Let $X_1, \dots, X_k \subseteq V(G)$ be an input to the colourful independence oracle of G, and let $X_{i,j} = X_i \cap V_j$ for all $i, j \in [k]$. Since G is k-partite with vertex classes $V_1, \dots, V_k$, every edge e of $G[X_1 \cup \dots \cup X_k]$ contains one vertex from each of $X_{1,\sigma_e(1)}, X_{2,\sigma_e(2)}, \dots, X_{k,\sigma_e(k)}$ for some map $\sigma_e \colon [k] \to [k]$. Moreover, e

is colourful under $X_1, \dots, X_k$ if and only if σ_e is a bijection. Motivated by this, for each bijection $\sigma : [k] \to [k]$, let $G_\sigma = G[X_{1,\sigma(1)} \cup \dots \cup X_{k,\sigma(k)}]$; then every colourful edge of G appears in exactly one G_σ, and every edge appearing in any G_σ is colourful. It follows that

$$\mathrm{cIND}_G(X_1, \dots, X_k) = \min\big\{\mathrm{IND}_G(X_{1,\sigma(1)} \cup \dots \cup X_{k,\sigma(k)}) \colon \sigma \text{ a bijection } [k] \to [k]\big\}.$$

We can compute each tuple $(X_{1,\sigma(1)}, \dots, X_{k,\sigma(k)})$ in $\mathcal{O}(n)$ time, each oracle invocation takes T time, and there are $k!$ such tuples to compute; thus overall we can compute $\mathrm{cIND}_G(X_1, \dots, X_k)$ in $(k \log n)^{\mathcal{O}(k)}(n + T)$ time as required. ∎

Finally, we give a formal proof of Theorem 1.7, which is straightforward from Corollary 1.3 and Definition 1.5.

THEOREM 1.7. *Let Π be a uniform witness problem, and let T be any function from instances of Π to the positive reals. Suppose that given an instance x of Π, there is an algorithm to solve* COLOURFUL-Π *on x with error probability at most $1/3$ in time $T(x)$. Suppose also that any such algorithm has running time $\widetilde{\Omega}(|x|)$. Then there is a randomised algorithm which, given an instance x of Π and $\varepsilon > 0$, with running time*

$$\varepsilon^{-2}(k_x \log n_x)^{\mathcal{O}(k_x)} \cdot \max\big\{T(I_x(S)) \colon S \subseteq V(G_x)\big\}, \tag{1.1}$$

outputs an ε-approximation to $\#\Pi(x)$ with probability at least $2/3$ and an ε-approximate sample from W_x with error probability zero.

Proof. Let x be an instance of Π, and let $\varepsilon > 0$. For all disjoint $X_1, \dots, X_{k_x} \subseteq V(G_x)$, we may compute $\mathrm{cIND}_{G_x}(X_1, \dots, X_{k_x})$ as follows:

(i) Compute $I_x(X_1 \cup \dots \cup X_{k_x})$ using the algorithm of Definition 1.5(iii);
(ii) Run the assumed COLOURFUL-Π algorithm on $y = I_x(X_1 \cup \dots \cup X_{k_x})$ with input partition $X_1, \dots, X_{k_x}$ of $G_y = G_x[X_1 \cup \dots \cup X_{k_x}]$.
(iii) Return 1 if COLOURFUL-Π returns `No` and 0 if COLOURFUL-Π returns `Yes`.

Denote this implementation by A. For brevity, let $T^+(x) = \max\big\{T(I_x(S)) \colon S \subseteq V(G_x)\big\}$; then A has running time $\widetilde{\mathcal{O}}(|x| + T^+(x))$. Since $T^+(x) \geq T(x) = \widetilde{\Omega}(|x|)$ by hypothesis, A therefore has running time $\widetilde{\mathcal{O}}(T^+(x))$.

Our algorithm now simply computes $V(G_x)$ and k_x from x using the algorithm of Definition 1.5(ii), then returns $\texttt{SampleCount}(V(G_x), k_x, A, \varepsilon, 2/3)$; by Corollary 1.3, this will yield an ε-approximation to $|E(G_x)|$ and the desired ε-approximate sample from $E(G_x) = W_x$ with the desired error probability. By Definition 1.5(i) we have $|E(G_x)| = \#\Pi(x)$, and $E(G_x) = W_x$ by definition, so the algorithm performs as required.

We now bound the running time. Observe from Definition 1.5(ii) that $n_x = \widetilde{\mathcal{O}}(|x|)$, and recall that by hypothesis we have $T^+(x) \geq T(x) = \widetilde{\Omega}(|x|)$; hence by Corollary 1.3, the overall running time is

$$\widetilde{\mathcal{O}}(|x|) + \varepsilon^{-2} \log^2(1/\delta)(k_x \log n_x)^{\widetilde{\mathcal{O}}(k)}(n_x + T^+(x)) \\ = \varepsilon^{-2} \log^2(1/\delta)(k_x \log n_x)^{\widetilde{\mathcal{O}}(k)} T^+(x),$$

as required. ∎

6.1. Application to Graph Motif. We now describe an approximate counting problem in parameterised complexity for which our results imply an immediate

improvement over the best stated running time bound in the literature: the Graph Motif problem, introduced by Lacroix, Fernandes and Sagot [40] 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. We write $k = |M|$. A *motif witness* for M is a set $U \subseteq V(G)$ of k vertices such that the induced subgraph $G[U]$ is connected and the colour multiset of U is exactly M. If M has a motif witness, then M is called a *motif*. Without loss of generality, let us assume that the set C of allowed colours always satisfies $C = \{1, \ldots, n\}$. The problem is formally stated as follows.

Graph Motif
Input: A graph G on n vertices and m edges, a colouring $c\colon V(G) \to C$, 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?

There has been substantial progress in recent years on improving the running-time of decision algorithms for Graph Motif [8, 13, 24, 31, 39]. Björklund, Kaski, and Kowalik [13] 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 6.1 (Björklund, Kaski, and Kowalik [13]). *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.*

For the counting version of Graph Motif, Guillemot and Sikora [31] addressed the related problem of counting k-vertex sub*trees* whose multiset of vertex colours equals M. This problem is equivalent to counting motif witnesses U for M weighted by the number of trees spanned by U. When M is a set, this exact counting problem admits an FPT algorithm, but is #W[1]-hard otherwise. Subsequently, Jerrum and Meeks [35] addressed the more natural counting analogue of Graph Motif in which the goal is to count all 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. The goal in [35] was simply to demonstrate the existence of an FPTRAS rather than to optimise the running time; the algorithm as described has running time $\Omega(n^3)$. We believe that, with sufficient care, the general strategy described in [35] could be adapted to give a running time similar to that obtained in the following corollary; however, Theorem 1.7 allows us to deduce this improvement immediately from Theorem 6.1. Moreover, it provides a method for translating any future improvement to the decision algorithm into an improved algorithm for approximate counting or sampling.

Corollary 6.2. *Given an n-vertex, m-edge instance (G, c, M) of* Graph Motif *with $k = |M|$ 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 $\varepsilon^{-2} k^{\mathcal{O}(k)} m \log^{\mathcal{O}(k)} n$.*

In order to apply Theorem 1.7, we need to show that Graph Motif is a uniform

witness problem according to Definition 1.5. Given an instance $x = (G, c, M)$, we let G_x be the k-hypergraph on the vertex-set $V(G)$ whose edges are all sets $U \subseteq V(G)$ that are motif witnesses for M. It is clear that G_x satisfies the conditions of Definition 1.5 on taking $I_x(S) = (G[S], c|_S, M)$, and so Graph Motif is a uniform witness problem. Another precondition of Theorem 1.7 is that any algorithm for the problem requires at least time $\widetilde{\Omega}(n)$; this is true for Graph Motif, because any algorithm for it must read a constant proportion of the input. Finally, before we can apply Theorem 1.7 to immediately obtain Corollary 6.2, we must state an algorithm for the problem Colourful-Graph Motif of Definition 1.6; to avoid confusion with the colouring that already exists, we instead call this Partitioned Graph Motif. It is clear that the maximum running time for this algorithm on any sub-instance will be dominated by that for the original instance.

Lemma 6.3. *There exists a randomised algorithm for* Partitioned Graph Motif *that runs in time* $k^{\mathcal{O}(k)}m$*, with error probability at most* $1/3$.

Proof. The input to Partitioned Graph Motif consists of (G, c, M) and a partition of $V(G)$ into disjoint sets $S_1, \ldots, S_k$. We now describe an algorithm that decides whether there is a motif witness $U \subseteq V(G)$ that intersects each set S_i in exactly one vertex.

Write $M = \{c_1, \ldots, c_k\}$. For each possible bijection $\pi : [k] \to [k]$ we will determine, with probability at least $1 - \frac{1}{3k!}$, whether there is a motif witness U with $U = \{u_1, \ldots, u_k\} \subseteq V(G)$ such that, for all $i \in [k]$, we have $u_i \in S_i$ and $c(u_i) = c_{\pi(i)}$. We achieve this by solving a new instance (G, c', M') of Graph Motif using the algorithm of Theorem 6.1: we use the same input graph, 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 $v \in V_i$. Moreover, we set $M' = \{(c_{\pi(1)}, 1), \ldots, (c_{\pi(k)}, k)\}$. We silently replace $C \times [k]$ with $[n]$ by discarding unused colours and renaming the rest. To achieve the claimed success probability, it suffices to call the randomised algorithm for Graph Motif a total of $\lceil 3k \log k \rceil$ times, returning `Yes` if any of the calls returns `Yes`, and `No` otherwise.

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 $k!$ choices of π is at least $2/3$, and in this case we output the correct answer.

In total, we invoke the randomised Graph Motif algorithm $k!\lceil 3k \log k \rceil$ times, so the total running time is $\mathcal{O}(k!k \log k \cdot 2^k k^2 m \log k \log\log k \log\log\log k) = k^{\mathcal{O}(k)}m$. ∎

Corollary 6.2 is now immediate from Lemma 6.3 and Theorem 1.7. (Recall that, for any k-colour m-edge instance $x = (G, c, M)$ of Graph Motif and any $S \subseteq V(G_x) = V(G)$, we have $I_x(S) = (G[S], c|_S, M)$; thus the maximum in the right-hand side of (1.1) is simply $k^{\mathcal{O}(k)}m$.)

6.2. Application to *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 [28]), and it has become an important problem in fine-grained complexity theory [51]. 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 [38] very recently developed almost linear-size linear decision trees for k-SUM, the fastest known algorithm for this problem still runs in time $\widetilde{\mathcal{O}}(n^{\lceil k/2 \rceil})$. A running time of $n^{o(k)}$ as $k \to \infty$ is ruled out under the exponential-time hypothesis [46]. We prove that any sufficiently non-trivial improvement over the best known decision algorithm carries over to approximate counting and witness sampling.

For any 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.

Observe that k-SUM is a uniform witness problem by Definition 1.5: The k-hypergraph has vertex set X, its edges are precisely the witnesses for X, and for all $S \subseteq X$, the instance $I_X(S)$ is simply S. Thus, the problem Colourful-k-SUM of Definition 1.6 is given as follows: The input consists of k disjoint sets $X_1, \dots, X_k$ of integers, and the goal is to determine whether there exist $x_1 \in X_1, \dots, x_k \in X_k$ such that $\sum_i x_i = 0$. Note that any algorithm for Colourful-k-SUM requires time $\widetilde{\Omega}(|X|)$ to read the input.

In order to apply Theorem 1.7, we first reduce Colourful k-SUM to k-SUM with the following lemma (which is well-known folklore).

Lemma 6.4. *If an n-integer instance of k-SUM can be solved in time $T(n)$, then there is an $\mathcal{O}(T(n))$-time algorithm for* Colourful k-SUM.

Proof. Let $X_1, \dots, X_k$ be the input for Colourful k-SUM. 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 $Y = Y_1 \cup \dots \cup Y_k$. Now we run the assumed k-SUM decision algorithm on Y and output the result. Preparing the set Y and running the algorithm takes time $\mathcal{O}(n) + T(n)$. Since any algorithm for k-SUM must read a constant proportion of its input, we have $T(n) \geq \Omega(n)$; thus, the overall running time of this algorithm is $\mathcal{O}(T(n))$.

To prove correctness, let $x_i \in X_i$ for all $i \in [k]$ such that $\sum_i x_i = 0$. Then

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

and so there are k distinct numbers in Y whose sum is zero. Conversely, suppose $y_1, \dots, y_k \in Y$ are distinct numbers whose sum is zero. 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, $\sum_i f^{-1}_{\sigma(i)}(y_i) = 0$. Thus $\{f^{-1}_{\sigma(1)}(y_1), \dots, f^{-1}_{\sigma(k)}(y_k)\}$ is a witness for Colourful k-SUM as required. We conclude that the reduction is correct. ∎

The following corollary is now immediate from Lemma 6.4 and Theorem 1.7. (Recall that, for any n-element instance X of k-SUM and any $S \subseteq V(G_X) = X$, we have $I_X(S) = S$; thus the maximum in the right-hand side of (1.1) is simply $T(n)$.)

Corollary 6.5. *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 $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(n))$.*

6.3. Application to k-Orthogonal Vectors. As is standard, we abbreviate k-Orthogonal vectors to k-OV. The k-OV problem has connections to central conjectures in fine-grained complexity theory [1, 30]. Clearly, k-OV can be solved in time $O(N^k D)$ using exhaustive search. Gao et al. [30] stated the Moderate-Dimension k-OV Conjecture, which says that k-OV cannot be solved in time $O(N^{k-\varepsilon}\operatorname{poly}(D))$ for any $\varepsilon > 0$. We show that every superlogarithmic improvement over exhaustive search for k-OV carries over to approximate counting and sampling of k-OV witnesses. Note that such an improvement is already known for 2-OV, namely 2-OV has an $N^{2-1/\mathcal{O}(\log(D/\log N))}$-time algorithm [3]. Chan and Williams [15] already generalised this 2-OV decision algorithm to an exact counting algorithm. For all integers $k \geq 2$, the k-OV problem is stated formally as follows.

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

We remark that the sum and product in the problem definition refer to the usual arithmetic operations over $\mathbb{Z}$. Observe that k-OV is a uniform witness problem according to Definition 1.5: for a k-OV instance $x = (X_1, \ldots, X_k)$, the k-hypergraph G_x has vertex-set $X_1 \cup \cdots \cup X_k$, its edges are precisely the witnesses of the k-OV instance, and for any $S \subset X_1 \cup \cdots \cup X_k$, the instance $I_x(S)$ is $(S \cap X_1, \ldots, S \cap X_k)$. Notice that k-OV is in fact a colourful uniform witness problem, since G_x is always k-partite, and so Colourful-k-OV reduces to $k^{\mathcal{O}(k)}$ instances of k-OV.

The following corollary is now immediate by Theorem 1.7, since any algorithm for k-OV requires time $\widetilde{\Omega}(|X_1 \cup \cdots \cup X_k|)$ to read the input and, by the definition of $I_x(S)$ and our choice of T, it is immediate that the maximum in the right-hand side of (1.1) is simply $T(x)$.

Corollary 6.6. *Fix $k \geq 2$, suppose an N-vector D-dimensional 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 $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(N, D))$.*

6.4. Application to first-order model checking. In this section, we apply our results to the *property testing problem* for k-FO. Informally, the input to this problem consists of 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. In order to formally introduce this problem, we must introduce some standard notation from logic; for the convenience of the reader, we will also give a brief introduction to the associated concepts.

Syntax. The class k-FO is the set of all first-order formulas φ in prenex normal form with at most k variables. Here is an example of a formula $\Phi \in 5$-FO:

$$\Phi(x_1, x_2, x_3) = \quad \forall x_4 \exists x_5 \,.\, R_1(x_1, x_2) \wedge (R_2(x_5, x_3, x_1) \Rightarrow R_1(x_4, x_5)) \,.$$

Because all the quantifiers are at the front, the formula is in *prenex normal form*. The formula Φ has *free* variables x_1, x_2, x_3 and uses two *relation symbols* R_1 and R_2. The *arity* of R_1 is 2 and the arity of R_2 is 3. Therefore, we say that Φ uses the *vocabulary*

$\nu_\Phi = (R_1, R_2, 2, 3)$.

In general, a *vocabulary* is a tuple $\nu = (R_1, \ldots, R_r, \alpha_1, \ldots, \alpha_r)$, where $R_1, \ldots, R_r$ are *relation symbols* and $\alpha_1, \ldots, \alpha_r$ are positive integers denoting the respective arities of the relation symbols. A formula $\varphi \in k$-FO over the vocabulary ν satisfies

$$\varphi(x_1, \ldots, x_\ell) = \quad Q_1 x_{\ell+1} Q_2 x_{\ell+2} \ldots Q_{k-\ell} x_k \,.\, \psi(x_1, \ldots, x_k)\,,$$

where the variables $x_1, \ldots, x_\ell$ are the *free* variables of φ, each $Q_i \in \{\exists, \forall\}$ is a quantifier, and ψ is a quantifier-free Boolean formula over the variables $x_1, \ldots, x_k$ such that each atom of ψ is of the form $R_i(x_{j_1}, \ldots, x_{j_{\alpha_i}})$, that is, each atom of ψ applies some relation symbol R_i to a tuple of α_i variables.

Semantics. So far, the formulas Φ and φ only consist of syntax. In order to be able to say that a formula is *satisfiable*, we need to define a *universe* U as well as instantiate the relation symbols with actual relations over U of matching arities. In our example, we might set the universe to be $U_\Phi = \{a, b, c, d\}$. Then the variables of Φ take values from U_Φ, and all quantification is over U_Φ. We might then instantiate the relations via $\mathcal{R}_1 = \{(a, b), (b, c)\}$ and $\mathcal{R}_2 = \{(a, d, c)\}$. We say that $\mathcal{S}_\Phi = (U, \mathcal{R}_1, \mathcal{R}_2)$ is a *structure* for ν_Φ. Now we can see that $\Phi(a, b, c)$ is true and $\Phi(a, c, d)$ is false in the structure $\mathcal{S}_\Phi$, and that Φ is *satisfiable* in $\mathcal{S}_\Phi$ because it has at least the satisfying assignment (a, b, c).

In general, a *structure* $\mathcal{S} = (U, \mathcal{R}_1, \ldots, \mathcal{R}_r)$ for the vocabulary ν consists of a finite *universe* U together with relations $\mathcal{R}_1, \ldots, \mathcal{R}_r$ over U such that the arity of each $\mathcal{R}_i$ is equal to α_i. An *assignment* $(y_1, \ldots, y_\ell) \in U^\ell$ to the free variables of φ is called *satisfying* in $\mathcal{S}$ if $\varphi(y_1, \ldots, y_\ell)$ is true in the structure $\mathcal{S}$. For all positive integers k, we define the following problem.

k-FO Property Testing
Input:
(i) A vocabulary $\nu = (R_1, \ldots, R_r, \alpha_1, \ldots, \alpha_r)$ with $\alpha_i \leq k$ for all i,
(ii) A structure $\mathcal{S} = (U, \mathcal{R}_1, \ldots, \mathcal{R}_r)$ on ν, where all tuples in all $\mathcal{R}_i$'s are explicitly given as a list and $\emptyset \neq \mathcal{R}_i \subseteq U^{\alpha_i}$ holds for all i,
(iii) A first-order formula φ in prenex normal form and with vocabulary ν, free variables $x_1, \ldots, x_\ell$, and at most $k - \ell$ quantifiers.
Question: Does φ have a satisfying assignment in $\mathcal{S}$?

Property testing is an important problem in logic and in database theory (cf. [23]). The fine-grained complexity of property testing has recently been studied to some extent [19, 30, 49]. In particular, Gao et al. [30] devise an algorithm for the property testing problem that runs in time $m^{k-1}/2^{\Theta(\sqrt{\log m})}$ for any fixed k-FO formula, where m is the number of distinct tuples in the input relations. This improves upon an already non-trivial $\widetilde{\mathcal{O}}(m^{k-1})$ time algorithm.

We prove the following reduction from approximate counting to decision.

Corollary 6.7. *Fix $k \in \mathbb{Z}_{\geq 0}$, and suppose k-FO Property Testing has an algorithm that runs in time $T(|\varphi|, n, m)$, where $|\varphi|$ is the size of the formula, n is the size of the universe, and m is the total number of tuples in the structure. Let $\mathcal{W}$ be the set of satisfying assignments. Then there is a randomised algorithm to ε-approximate $|\mathcal{W}|$, or draw an ε-approximate sample from $\mathcal{W}$, in time $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(|\varphi| + \ell, n, m + n))$.*

This corollary follows from Theorem 1.7 by using the assumed algorithm for k-FO Property Testing to simulate the uncoloured independence oracle of an appropri-

ate hypergraph.

Proof. In order to apply Theorem 1.7 to property testing, we need to show how k-FO Property Testing is a uniform witness problem according to Definition 1.5. Let $x = (\nu, \mathcal{S}, \varphi)$ be an instance of k-FO Property Testing, let $\mathcal{S} = (U, \mathcal{R}_1, \dots, \mathcal{R}_r)$, and let ℓ with $\ell \leq k$ be the number of free variables in φ. We define the hypergraph $G = G_x$ as follows: For all $i \in \{1, \dots, \ell\}$, let $U_i = U \times \{i\}$. Define an ℓ-hypergraph G with vertex set $U_1 \cup \dots \cup U_\ell$ whose edges are the sets $\{(y_1, 1), \dots, (y_\ell, \ell)\}$ such that $(y_1, \dots, y_\ell)$ is a satisfying assignment of φ in $\mathcal{S}$.

It is clear that this function $x \mapsto G_x$ satisfies the conditions (i) and (ii) of Definition 1.5. To prove that k-FO Property Testing is a uniform witness problem, it remains to prove (iii). Indeed, let $X \subseteq V(G)$ be given as an input in addition to x. Then we prepare in time $\widetilde{\mathcal{O}}(|x|)$ an instance $I_x(X) = (\nu', \mathcal{S}', \varphi')$ of k-FO Property Testing such that $G_{I_x(X)} = G_x[X]$ holds: We set $X_i = X \cap U_i$ for all $i \in \{1, \dots, \ell\}$. We form the new instance $(\nu', \mathcal{S}', \varphi')$ of k-FO Property Testing as follows: form ν' from ν by adding ℓ additional relation symbols $\in_{X_1}, \dots, \in_{X_\ell}$, each with arity 1; form $\mathcal{S}'$ from $\mathcal{S}$ by instantiating each $\in_{X_i}$ with the set $\{y \colon (y, i) \in X_i\}$; and form φ' from φ by conjoining it with the formula $(\in_{X_1}(x_1) \wedge \dots \wedge \in_{X_\ell}(x_\ell))$. Then $G_{I_x(X)} = G_x[X]$ indeed holds. Thus, (iii) holds and k-FO Property Testing is a uniform witness problem. In fact, since all hypergraphs involved are k-partite, it is a colourful uniform witness problem.

A technical precondition of Theorem 1.7 is that any algorithm for the problem requires at least time $\widetilde{\Omega}(|x|)$; this is true for k-FO Property Testing, because any algorithm for it must read a constant proportion of the input. Since k-FO Property Testing is a colourful uniform witness problem, we obtain an algorithm for Colourful-Π with an overhead of only $k^{\mathcal{O}(k)} = \mathcal{O}(1)$ time. Thus, we can apply the theorem and obtain an algorithm for ε-approximate counting and sampling that runs in time

$$\varepsilon^{-2}(k \log n)^{\mathcal{O}(k)} k^{\mathcal{O}(1)} \cdot \max_{X \subseteq V(G)} T(I_x(X)) \,.$$

Since all instances $I_x(X)$ have a universe of size n, at most n additional tuples, and a formula φ' that is φ conjoined with ℓ additional terms, we have $T(I_x(X)) \leq T(|\varphi| + \ell, n, m + n)$. The claimed running time then follows from $k = \mathcal{O}(1)$. ∎

By applying Corollary 6.7, we are able to lift the algorithm of Gao et al. [30], to approximate counting and sampling. For example, we obtain an algorithm to ε-approximately sample uniformly random satisfying assignments. For any fixed k-FO formula φ, this algorithm runs in time $\varepsilon^{-2} m^{k-1} / 2^{\Theta(\sqrt{\log m})}$. We remark the following subtleties: Gao et al. [30] state their algorithm for any fixed formula φ, but actually their algorithm is uniform in φ with a running time that can be expressed as a product $f(|\varphi|) \cdot T(n, m)$. This means that increasing the size of φ by a constant does not change the asymptotic running time for any fixed φ, since $f(|\varphi| + \ell)$ is constant as well. Finally, for certain quantifier structures of φ, they obtain even faster decision algorithms; since our reduction does not change the quantifier structure, these running times transfer to approximate counting and sampling as well.

6.5. Application to subgraph problems. Recall from the discussion following the statement of Theorem 1.1 that the theorem can be applied to the problem #k-Clique for every constant k. Thus, every $T(n)$-time algorithm to decide the existence of a k-clique can be turned into a $\widetilde{\mathcal{O}}(\varepsilon^{-2} T(n))$-time algorithm to ε-approximate the number of k-cliques. We now generalise this result to the problems of finding

colourful or zero-weight copies of an arbitrary subgraph, and we obtain similar consequences for approximately uniform sampling via Corollary 1.4.

6.5.1. Colourful subgraphs. In this section, we use Corollary 1.4 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 in adjacency list format and a vertex-colouring $c : V(G) \to [k]$ (not necessarily proper).
> *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 $c(v) = i$?

Díaz, Serna, and Thilikos [20] use dynamic programming to solve Colourful-H in time $\widetilde{\mathcal{O}}(n^{t+1})$, where t is the treewidth of H — their algorithm works for the exact counting version of the problem, too. Marx [42] asks, loosely speaking, whether the decision problem can be solved in time $n^{o(t)}$, and proves that $n^{o(t/\log t)}$ is impossible under the exponential-time hypothesis (ETH). For all constant $\gamma > 0$, we show that a factor n^{γ} improvement to Díaz et al.'s algorithm for the decision problem would immediately yield a corresponding improvement to the approximate counting or sampling problems.

Corollary 6.8. *Let H be any fixed graph. Suppose n-vertex m-edge instances of Colourful-H can be solved by a randomised algorithm 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 $\varepsilon^{-2} \cdot \widetilde{\mathcal{O}}(T(m, n))$.*

In the proof, we use the notion of a graph homomorphism (see [41]). A function $f : V(H) \to V(G)$ is called a *homomorphism* if, for all $\{u, v\} \in E(H)$, we have $\{f(u), f(v)\} \in E(G)$. Note that injective homomorphisms correspond to subgraph embeddings.

Proof. Let (G, c) be an instance of Colourful-H. To motivate our proof, we first discuss a natural approach which fails. First, remove all edges $\{u, v\} \in E(G)$ with $c(u) = c(v)$ from the graph, and note that this makes G k-partite with vertex classes $c^{-1}(1), \ldots, c^{-1}(k)$ but does not change the answer. It would be natural to apply Theorem 1.7 to the k-partite k-hypergraph $\mathcal{G}$ on $V(G)$ in which $S \subseteq V(G)$ is an edge of $\mathcal{G}$ if $G[S]$ contains a subgraph isomorphic to H, and indeed when H is a k-clique this approach works. However, in general each edge S of $\mathcal{G}$ may correspond to more than one copy of H contained in $G[S]$. Instead, we will apply Corollary 1.4 directly to count the edges in multiple k-hypergraphs corresponding to the $k!$ possible ways H could be embedded as a subgraph in $G[S]$.

For each bijective function $d : V(H) \to [k]$, we let $\mathcal{G}_d$ be a k-hypergraph with vertex set $V(G)$ and edges given as follows. For each size-k subset $S \subseteq V(G)$ with the property that $c|_S : S \to [k]$ is bijective (i.e. S is colourful under c), we add S to $E(\mathcal{G}_d)$ 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

$$|\mathcal{H}| = \frac{1}{\mathrm{Aut}(H)} \sum_d e(\mathcal{G}_d), \tag{6.1}$$

where $\mathrm{Aut}(H)$ is the number of automorphisms of H. Observe that since H is fixed,

we have access to $\mathrm{Aut}(H)$; thus given the claim, the problem of ε-approximating $|\mathcal{H}|$ reduces to that of ε-approximating each term $e(\mathcal{G}_d)$.

Proof of (6.1): We proceed by double-counting the number N of injective homomorphisms $h\colon V(H) \to V(G)$ such that the image $h(H)$ is colourful in G with respect to c, i.e. the number of ways H can be embedded in G as a colourful subgraph. On the one hand, each such h has a unique image $S = h(V(H)) \subseteq V(G)$ and a unique labelling $d\colon V(H) \to [k]$ such that $h = c|_S^{-1} \circ d$; thus it corresponds bijectively to the edge spanning S in $\mathcal{G}_d$, and $\sum_d e(\mathcal{G}_d) = N$. On the other hand, each copy of H in G corresponds to $\mathrm{Aut}(H)$ embeddings of H into G, so we have $|\mathcal{H}|\mathrm{Aut}(H) = N$. Combining these two equations yields (6.1) as required.

Approximating the terms: We will now ε-approximate each term $e(\mathcal{G}_d)$ of the right-hand side of (6.1) using `PartitionedSampleCount` from Corollary 1.4 with $\delta = 1/(3k!)$. By a union bound, the probability that at least one of the $k!$ invocations of `PartitionedSampleCount` fails is at most $1/3$. To approximate $e(\mathcal{G}_d)$ using `PartitionedSampleCount`, we first recall that $\mathcal{G}_d$ is a k-partite k-hypergraph with explicitly given vertex classes, and then implement the uncoloured independence oracle for $\mathcal{G}_d$ using the assumed randomised decision algorithm for Colourful-H. To this end, we define the following algorithm $A(X)$:

1. On input $X \subseteq V(G)$, the algorithm constructs a graph G' on the vertex set $V(G') = X$. Start with $G[X]$ and delete all edges internal to colour classes and all edges between colour classes whose corresponding vertices are not joined in H. (Conversely, we only add $\{u, v\} \in E(G[X])$ to $E(G')$ if $\{d^{-1}(c(u)), d^{-1}(c(v))\} \in E(H)$ holds.)
2. Run the assumed algorithm for Colourful-H on input (G', c') where $c' = c|_{V(G')}$ and return its output.

Note that A runs in time $\widetilde{\mathcal{O}}(n+m+T(m,n))$. 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 A in fact runs in time $\widetilde{\mathcal{O}}(T(n,m))$. If A implements the uncoloured independence oracle for $\mathcal{G}_d$ with error probability at most $1/3$, we can pass the description of this algorithm along with the vertex classes $X_i = X \cap c^{-1}(i)$ for all $i \in [k]$ and parameters k, ε, and $\delta = 1/(3k!)$ to `PartitionedSampleCount`. Our running time bounds therefore follow, and it remains to prove correctness.

Correctness: First assume that $A(X)$ accepts, so that the goal is to show $e(\mathcal{G}_d[X]) > 0$. Because A accepts, there is an H-isomorphic colourful subgraph F of (G', c). Let $S = V(F)$. By the construction of G', all edges in F must be between colour classes whose corresponding vertices are joined in H, so the function $c|_S^{-1} \circ d$ is an injective homomorphism from H to $G'[S]$. Thus $S \in E(\mathcal{G}_d[X])$, so $e(\mathcal{G}_d[X]) > 0$ as required.

For the reverse direction, suppose $e(\mathcal{G}_d[X]) > 0$. We need to show that $A(X)$ accepts with probability at least $2/3$. Let $S \in E(\mathcal{G}_d[X])$. Then by the definition of $\mathcal{G}_d$, the function $c|_S^{-1} \circ d$ 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. In this case, our assumed algorithm for Colourful-H, and hence also $A(X)$, accepts with probability at least $2/3$. We have proved correctness as required.

Sampling: Our sampling algorithm is very similar to our approximate counting algorithm, again using the fact that each subgraph corresponds to $\mathrm{Aut}(H)$ embeddings, so we omit the details. The only differences are the following:

1. We require that each invocation of our approximate counting algorithm has

failure probability at most $\varepsilon/(10k^{2k}\ln(5/\varepsilon))$ and set δ accordingly.

2. Rather than outputting a weighted sum of ε-approximations of each $e(\mathcal{G}_d)$, we use $(\varepsilon/10)$-approximations of each $e(\mathcal{G}_d)$ to apply rejection sampling as in e.g. Florescu [25, Proposition 3.3]. Thus writing N for our $(\varepsilon/10)$-approximation of $|\mathcal{H}|$ and N_d for our $(\varepsilon/10)$-approximation of $e(\mathcal{G}_d)$, we choose a bijective function $d\colon [k] \to [k]$ uniformly at random, take an $(\varepsilon/10)$-approximate sample from $E(\mathcal{G}_d)$, and accept and output this sample with probability N_d/N; otherwise, we reject it and resample. If we require more than $k^{2k}\ln(5/\varepsilon)$ iterations, we return an arbitrary output.

Note that the acceptance probability at each step is at least $1/(\mathrm{Aut}(H)^2) \geq 1/k^{2k}$, and acceptance is independent at each step, so the probability we require more than $k^{2k}\ln(5/\varepsilon)$ invocations is at most $(1-1/k^{2k})^{k^{2k}\ln(5/\varepsilon)} \leq \varepsilon/5$. Combining this with the error in the rejection sampling from our $(\varepsilon/10)$-approximations, and with the probability that our approximate counting algorithm fails to return a correct value in the first $k^{2k}\ln(5/\varepsilon)$ invocations (which is at most $\varepsilon/10$ by a union bound), we see that our algorithm returns an ε-approximate sample as required. Since $\varepsilon \geq n^{-k}$, the running time bound is immediate. ∎

6.5.2. Weighted subgraphs. In an edge-weighted graph, the graph G is augmented with a function $w : E(G) \to \mathbb{Z}$. The weight $w(F)$ of a subgraph F of G is the sum $\sum_{e\in E(F)} w(e)$ of all edge-weights in F. We now consider the following computational problem for any fixed unweighted graph H.

> Exact-Weight-H
> *Input:* An edge-weighted graph G with (perhaps negative) integer weights.
> *Question:* Does G have a subgraph isomorphic to H with total weight zero?

The special case where H is a k-clique has been studied in fine-grained complexity under the name Exact-Weight k-Clique. It has been conjectured [1] that there does not exist any real $\varepsilon > 0$ and integer $k \geq 3$ such that the Exact-Weight k-Clique problem on n-vertex graphs and with edge-weights in $\{-M, \ldots, M\}$ can be solved in time $n^{(1-\varepsilon)k}\,\mathrm{polylog}(M)$. For the closely related Min-Weight k-Clique problem, only subpolynomial-time improvement over the exhaustive search algorithm is known [1, 50, 15], with a running time of $n^k/\exp(\Omega(\sqrt{\log n}))$. As we now show, Theorem 1.7 implies 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 6.9. *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 $\varepsilon^{-2}\cdot\widetilde{\mathcal{O}}(T(n, m, M))$.*

Proof. First observe that Exact-Weight k-Clique is a uniform witness problem by Definition 1.5: Given an instance (G, w, k) of Exact-Weight k-Clique, we have $V(G_x) = V(G)$ and the edges of G_x are precisely the k-cliques of G, with $I_{(G,w,k)}(S) = (G[S], w|_S, k)$ for all $S \subseteq V(G)$. Moreover, any randomised algorithm for Colourful-Exact-Weight k-Clique (see Definition 1.6) must read at least a constant proportion of the input bits, and so the lower bound on its running time required by Theorem 1.7 is satisfied. Hence, by Theorem 1.7, it suffices to give an algorithm for Colourful-Exact-Weight k-Clique with running time $\widetilde{\mathcal{O}}(T(n, m, M))$.

Let (G, w, k) be an instance of Exact-Weight k-Clique, and let $X_1, \ldots, X_k$

be a partition of $V(G)$. Then we form a graph G' from G in linear time by removing all edges internal to each vertex set X_i, and apply our Exact-Weight k-Clique decision algorithm to G' in time at most $T(n, m, M)$. The cliques of G' are precisely the colourful cliques of G with respect to $V_1, \ldots, V_k$, so this solves Colourful-Exact-Weight k-Clique in $\tilde{\mathcal{O}}(m + n + T(n, m, M))$ time as required. ∎

There is a generalisation of Exact-Weight k-Clique to edge-weighted d-hypergraphs, for which a fine-grained complexity conjecture exists [1]. A result analogous to Corollary 6.9 holds for this problem as well, but we do not state it formally.

Instead, we now state a more interesting corollary for Exact-Weight-H. When H is a k-clique, it was sufficient in the proof of Corollary 6.9 to delete some edges to make sure that the colourful independence oracle only counts *colourful* copies of H. For general graphs H, we control this by relying on basic bit tricks that are commonly used in subset sum or exact-weight type problems. Unfortunately, we do not know how to do so inside the colourful independence oracle, at least not in the way in which we have formalised it. Instead, we perform an additional colour-coding step before running the algorithm of Corollary 1.4 as a black box, which leads to an additional ε^{-2} factor overhead in the running time. However, we believe that this is merely an artefact that can be avoided by doing some surgery on the proofs of our main results. Since doing so would be lengthy, and does not add any insight, we accept the loss in the running time to get a cleaner proof.

Corollary 6.10. *Let H be any fixed graph with k vertices. Suppose that n-vertex m-edge instances G 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 subgraphs of G that are isomorphic to H. Then there is a randomised algorithm to ε-approximate $|\mathcal{S}|$, or draw an ε-approximate sample from $\mathcal{S}$, in time $\varepsilon^{-4} \cdot \tilde{\mathcal{O}}(T(m, n, k^2 M))$.*

Proof. We first 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 $k = |V(H)|$. If $n < k$, we return the correct answer 0. If H has exactly i isolated vertices, we remove them from H, produce an estimate for the reduced graph, and multiply this estimate by $\binom{n-k+i}{i}$ to obtain the estimate for H. Thus, we can assume without loss of generality that $n \geq k$ and that H has no isolated vertices. The algorithm now proceeds as follows:

1. Let $t = \varepsilon^{-2} \cdot 100e^{2k}$. For all r from 1 to t:
   (a) Sample a partition $V_1 \cup \cdots \cup V_k = V(G)$ uniformly at random.
   (b) For all bijective functions $\pi : [k] \to V(H)$:
      i. Define the graph $G_{r,\pi}$ by deleting from G all edges $E(V_i, V_j)$ between parts V_i and V_j for which $\pi(i)\pi(j)$ is not an edge of H.
      ii. Define the k-partite k-hypergraph $\mathcal{G}_{r,\pi}$ with vertex set $V(G)$ such that each set $S \subseteq V(G)$ is an edge of $\mathcal{G}_{r,\pi}$ if and only if $G_{r,\pi}[S]$ is isomorphic to H, contains one vertex from each V_i, and is zero-weight with respect to w.
      iii. Call the algorithm `PartitionedSampleCount` from Corollary 1.4 on $\mathcal{G}_{r,\pi}$ with $\delta = 1/(100tk!)$ and with error parameter $\varepsilon/3$ to obtain an estimate $N_{r,\pi}$ for the number of edges in $\mathcal{G}_{r,\pi}$.
2. Output the estimate $\sum_{r,\pi} N_{r,\pi} \cdot k^k/(tk! \cdot \mathrm{Aut}(H))$.

Here the scaling factor $k^k/(tk! \cdot \mathrm{Aut}(H))$ arises from the fact that a random k-colouring of a set of size k is colourful with probability $k!/k^k$, each edge in our hypergraph corresponds to $\mathrm{Aut}(H)$ injective homomorphisms from H to G, and we repeat the

whole process t times.

We will explain how to simulate the uncoloured independence oracles of the hypergraphs $\mathcal{G}_{r,\pi}$ (as required by `PartitionedSampleCount`) shortly. First, we argue for the correctness of the algorithm.

Proving correctness given uncoloured independence oracles: `Count` is called $tk!$ times, and each invocation fails to produce an $(\varepsilon/3)$-approximation with probability at most δ. By a union bound, this implies that, with probability at least 0.99, *all* estimates $N_{r,\pi}$ are $(\varepsilon/3)$-approximations to the respective numbers $e(\mathcal{G}_{r,\pi})$ of edges in the hypergraphs $\mathcal{G}_{r,\pi}$. Conditioned on this event, the sum that is produced as output is thus an $(\varepsilon/3)$-approximation to $\sum_{r,\pi} e(\mathcal{G}_{r,\pi}) \cdot k^k/(tk! \cdot \mathrm{Aut}(H))$. Let s_r be the number of subgraphs of G that are isomorphic to H, are zero-weight with respect to w, and are colourful with respect to $V_1, \dots, V_k$. As in the proof of Corollary 6.8, each colourful zero-weight copy of H corresponds to exactly $\mathrm{Aut}(H)$ terms in $\sum_\pi e(\mathcal{G}_{r,\pi})$, and so the algorithm produces an $\varepsilon/3$-approximation to $\sum_r s_r k^k/(tk!)$.

We now prove that $\sum_r s_r k^k/(tk!)$ is likely to be an $(\varepsilon/3)$-approximation to the total number $|\mathcal{S}|$ of zero-weight subgraphs of G that are isomorphic to H. To do so, we first show that it is equal to $|\mathcal{S}|$ in expectation and then apply Hoeffding's inequality to prove concentration. For any zero-weight subgraph F of G that is isomorphic to H, let $X_{r,F} \in \{0,1\}$ be the indicator random variable for the event that F is a subgraph of $G[V_1, \dots, V_k]$ in the r'th iteration (that is, the event that there is some π with $V(F) \in E(\mathcal{G}_{r,\pi})$). Then $\mathbb{E}(X_{r,F}) = k!/k^k$, so we have

$$\mathbb{E}\Big(\sum_{r=1}^{t} s_r \frac{k^k}{tk!}\Big) = \mathbb{E}\Big(\sum_{r=1}^{t}\sum_{F\in\mathcal{S}} X_{r,F}\frac{k^k}{tk!}\Big) = \sum_{r=1}^{t}\sum_{F\in\mathcal{S}}\frac{1}{t} = |\mathcal{S}|.$$

Since each s_r lies in $[0, |\mathcal{S}|]$, it follows by Hoeffding's inequality (Lemma 2.1) that

$$\begin{aligned}\mathbb{P}\Big(\Big|\frac{k^k}{tk!}\sum_r s_r - |\mathcal{S}|\Big| > \frac{\varepsilon}{3}|\mathcal{S}|\Big) &= \mathbb{P}\Big(\Big|\sum_r s_r - \frac{tk!\cdot|\mathcal{S}|}{k^k}\Big| > \varepsilon\frac{tk!}{3k^k}|\mathcal{S}|\Big)\\ &\le 2\exp\left(-2\varepsilon^2\Big(\frac{tk!}{3k^k}|\mathcal{S}|\Big)^2/t|\mathcal{S}|^2\right)\\ &= 2\exp\big(-2\varepsilon^2 t(k!/3k^k)^2\big).\end{aligned}$$

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

$$\mathbb{P}\Big(\Big|\frac{k^k}{tk!}\sum_r s_r - |\mathcal{S}|\Big| > \frac{\varepsilon}{3}|\mathcal{S}|\Big) \le 2\exp\big(-t\varepsilon^2 e^{-2k}/9\big) \le 1/10.$$

Thus, with overall probability at least $4/5$, we have $(1-\frac{\varepsilon}{3})|\mathcal{S}| \le \frac{k^k}{tk!}\sum_r s_r \le (1+\frac{\varepsilon}{3})|\mathcal{S}|$.

We now note that for all $0 < \varepsilon < 1$, $(1-\varepsilon/3)^2 > 1-\varepsilon$ and $(1+\varepsilon/3)^2 < 1+\varepsilon$; hence an $(\varepsilon/3)$-approximation of an $(\varepsilon/3)$-approximation is an ε-approximation. It therefore follows by a union bound that with probability at least $2/3$, the output of our algorithm is an ε-approximation to $|\mathcal{S}|$. This completes our correctness analysis.

Implementing the uncoloured independence oracles: For each hypergraph $\mathcal{G}_{r,\pi}$, we will implement the uncoloured independence oracle using our assumed decision algorithm for Exact-Weight-H. Let $X \subseteq V(G)$; then we must determine whether $\mathcal{G}_{r,\pi}[X]$ contains an edge, i.e. whether $G_{r,\pi}[X]$ contains a subgraph which is isomorphic to H, is zero-weight with respect to w, and is colourful with respect to the partition $V_1, \dots, V_k$. We cannot simply apply the decision algorithm to $G_{r,\pi}[X]$ with

the same weight function w, because this algorithm may find zero-weight isomorphic copies of H that are not colourful. To deal with this issue, we give $G_{r,\pi}$ a new weight function $w_{r,\pi}$ which will enforce colourfulness using standard bit tricks.

For each $i \in [k]$, let $X_i = X \cap V_i$. Let h be the number of edges of H. We define the function $\iota : E(G) \to [h]$ as follows: We set $\iota(\{u,v\}) = \iota$ for all $\{u,v\} \in E(G)$ with $u \in V_i$ and $v \in V_j$ such that $\{\pi(i), \pi(j)\} \in E(H)$ is the ι'th edge of H in lexicographic order. We define the edge weights of $G_{r,\pi}$ by $w_{r,\pi}(e) = w(e) + \delta_{r,\pi}(e)$ for all e, where

$$\delta_{r,\pi}(e) = \begin{cases} (hM+1) \cdot (2h)^{\iota(e)} & \text{if } \iota(e) > 1; \\ -(hM+1) \cdot \sum_{\iota=2}^{h} (2h)^{\iota} & \text{if } \iota(e) = 1. \end{cases}$$

Let ϕ be the canonical homomorphism from $G_{r,\pi}$ to H, that is, the function that maps vertices from V_i to the vertex $\pi(i)$ in H.

We claim that for any subgraph F of $G_{r,\pi}$ isomorphic to H, we have $w_{r,\pi}(F) = 0$ if and only $w(F) = 0$ and F is colourful with respect to the partition $V_1, \dots, V_k$. If this is true, then we can simulate the uncoloured independence oracle of $\mathcal{G}_{r,\pi}$ applied to a set X by applying our decision algorithm to $G_{r,\pi}[X]$.

To prove the claim, first note that

$$w_{r,\pi}(F) = \sum_{f \in E(F)} \big(w(f) + \delta_{r,\pi}(f)\big) = w(F) + \sum_{f \in E(F)} \delta_{r,\pi}(f). \tag{6.2}$$

This latter sum has exactly h summands, and each summand is either $(hM+1)(2h)^{\iota}$ for some $\iota \geq 2$, or it is $-(hM+1)\sum_{\iota=2}^{h}(2h)^{\iota}$. If F is colourful, then ϕ maps every edge of F to a distinct edge of H, so the summands that occur are all different and trivially cancel; we therefore have $w_{r,\pi}(F) = w(F)$ in this case.

Suppose instead that F is not colourful, so that ϕ maps two distinct vertices x and y of F to the same vertex z of H. Since ϕ is a homomorphism, x and y are each joined to $d_H(z)$ colour classes out of $\{V_1, \dots, V_k\}$ in F; moreover, by the construction of $G_{r,\pi}$, at most $d_H(z)$ of these classes are distinct. Since H does not contain isolated vertices, we have $d_H(z) > 0$ and so some colour class must appear in both x's neighbourhood and y's neighbourhood. In other words, ϕ must map two different edges f, f' of F to the same edge e of H; this implies there is some $\iota \in [h]$ such that ϕ maps no edges to the ι'th edge of H. Thus, the summands on the right-hand side of (6.2) do not cancel, since for all $\iota \in [h]$ we have $h \cdot (2h)^{\iota} < (2h)^{\iota+1}$ and $(2h)^{\iota+1} - h \cdot (2h)^{\iota} > (2h)^{\iota}$, and so $w_{r,\pi} \neq 0$ as required. We have therefore successfully implemented the uncoloured independence oracle for $\mathcal{G}_{r,\pi}$, the missing ingredient in our algorithm.

Bounding the running time: Our algorithm creates $tk! = O(\varepsilon^{-2})$ weighted graphs $G_{r,\pi}$ and calls `PartitionedSampleCount` once for each. By Corollary 1.4, this takes time $\widetilde{\mathcal{O}}(\varepsilon^{-4}(m+n))$ and makes $\widetilde{\mathcal{O}}(\varepsilon^{-4})$ queries to the independence oracle. Each query to the independence oracle takes time $T(n, m, k^2M) = \Omega(n+m)$. Combining these facts, we obtain an overall running time of $\widetilde{\mathcal{O}}(\varepsilon^{-4}T(n, m, k^2M))$ as claimed.

Sampling algorithm: Our sampling algorithm is very similar to our approximate counting algorithm, so we omit the details. The only differences are the following.

1. We take only a single uniformly random partition $V_1 \cup \dots \cup V_k$ with associated graphs G_π for every $\pi \colon [k] \to V(H)$.
2. We require that each invocation of our approximate counting algorithm has failure probability at most $\varepsilon/(\ln(5/\varepsilon)10k^{2k})$ and set δ accordingly.
3. Rather than outputting a weighted sum of ε-approximations of each $e(G_\pi)$, we use $(\varepsilon/10)$-approximations of each $e(G_\pi)$ to apply rejection sampling

as in e.g. Florescu [25, Proposition 3.3]. Thus writing N for our $(\varepsilon/10)$-approximation of $e(G)$ and N_π for our $(\varepsilon/10)$-approximation of $e(G_\pi)$, we choose a bijective function $\pi\colon [k] \to [k]$ uniformly at random, take an $(\varepsilon/10)$-approximate sample from $E(G_\pi)$ using the `PartitionedSampleCount` algorithm from Corollary 1.4, and accept and output this sample with probability N_π/N; otherwise, we reject it and resample. If we require more than $k^{2k}\ln(5/\varepsilon)$ iterations, we return an arbitrary output.

Note that the acceptance probability at each step is at least $1/k^{2k}$, and acceptance is independent at each step, so the probability we require more than $k^{2k}\ln(5/\varepsilon)$ invocations is at most $(1-1/k^{2k})^{k^{2k}\ln(5/\varepsilon)} \leq \varepsilon/5$. Combining this with the error in the rejection sampling from our $(\varepsilon/10)$-approximations, and with the probability that our approximate counting algorithm fails to return a correct value in the first $k^{2k}\ln(5/\varepsilon)$ invocations (which is at most $\varepsilon/10$ by a union bound), we see that our algorithm returns an ε-approximate sample as required. Since $\varepsilon \geq n^{-k}$, the running time bound is immediate. ∎

Corollary 6.10 can be combined with known, non-trivial decision algorithms for Exact-Weight-H. For example, 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. We obtain the following.

Corollary 6.11. *Let H be a fixed graph. When $\mathcal{S}$ denotes the set of zero-weight H-subgraphs of a given n-vertex graph G, there is a randomised algorithm to ε-approximate $|\mathcal{S}|$, or draw an ε-approximate sample from $\mathcal{S}$, in time $\widetilde{\mathcal{O}}(\varepsilon^{-4}n^{\gamma(H)})$.*

REFERENCES

[1] A. Abboud, K. Bringmann, H. Dell, and J. Nederlof, *More consequences of falsifying SETH and the orthogonal vectors conjecture*, in Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018, I. Diakonikolas, D. Kempe, and M. Henzinger, eds., ACM, 2018, pp. 253–266, https://doi.org/10.1145/3188745.3188938.

[2] A. Abboud and K. Lewi, *Exact weight subgraphs and the k-sum conjecture*, in Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I, F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, eds., vol. 7965 of Lecture Notes in Computer Science, Springer, 2013, pp. 1–12, https://doi.org/10.1007/978-3-642-39206-1_1.

[3] A. Abboud, R. R. Williams, and H. Yu, *More applications of the polynomial method to algorithm design*, in Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015, P. Indyk, ed., SIAM, 2015, pp. 218–230, https://doi.org/10.1137/1.9781611973730.17.

[4] N. Alon and S. Gutner, *Balanced families of perfect hash functions and their applications*, ACM Trans. Algorithms, 6 (2010), pp. 54:1–54:12, https://doi.org/10.1145/1798596.1798607, https://doi.org/10.1145/1798596.1798607.

[5] N. Alon, R. Yuster, and U. Zwick, *Color-coding*, J. ACM, 42 (1995), pp. 844–856, https://doi.org/10.1145/210332.210337.

[6] V. Arvind and V. Raman, *Approximation algorithms for some parameterized counting problems*, in Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings, P. Bose and P. Morin, eds., vol. 2518 of Lecture Notes in Computer Science, Springer, 2002, pp. 453–464, https://doi.org/10.1007/3-540-36136-7_40.

[7] P. Beame, S. Har-Peled, S. N. Ramamoorthy, C. Rashtchian, and M. Sinha, *Edge estimation with independent set oracles*, ACM Trans. Algorithms, 16 (2020), pp. 52:1–52:27, https://doi.org/10.1145/3404867.

[8] N. Betzler, R. van Bevern, M. R. Fellows, C. Komusiewicz, and R. Niedermeier, *Pa-*

rameterized algorithmics for finding connected motifs in biological networks, IEEE ACM Trans. Comput. Biol. Bioinform., 8 (2011), pp. 1296–1308, https://doi.org/10.1109/TCBB.2011.19.

[9] A. Bhattacharya, A. Bishnu, A. Ghosh, and G. Mishra, *Hyperedge estimation using polylogarithmic subset queries*, CoRR, abs/1908.04196 (2019), http://arxiv.org/abs/1908.04196.

[10] A. Bhattacharya, A. Bishnu, A. Ghosh, and G. Mishra, *Triangle estimation using tripartite independent set queries*, in 30th International Symposium on Algorithms and Computation, ISAAC 2019, December 8-11, 2019, Shanghai University of Finance and Economics, Shanghai, China, P. Lu and G. Zhang, eds., vol. 149 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 19:1–19:17, https://doi.org/10.4230/LIPIcs.ISAAC.2019.19.

[11] A. Bhattacharya, A. Bishnu, A. Ghosh, and G. Mishra, *On triangle estimation using tripartite independent set queries*, Theory of Computing Systems, (2021), pp. 1166–1192.

[12] A. Bishnu, A. Ghosh, S. Kolay, G. Mishra, and S. Saurabh, *Parameterized query complexity of hitting set using stability of sunflowers*, in 29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan, W. Hsu, D. Lee, and C. Liao, eds., vol. 123 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, pp. 25:1–25:12, https://doi.org/10.4230/LIPIcs.ISAAC.2018.25.

[13] A. Björklund, P. Kaski, and Ł. Kowalik, *Constrained multilinear detection and generalized graph motifs*, Algorithmica, 74 (2016), pp. 947–967, https://doi.org/10.1007/s00453-015-9981-1.

[14] S. Boucheron, G. Lugosi, and P. Massart, *Concentration Inequalities - A Nonasymptotic Theory of Independence*, Oxford University Press, 2013, https://doi.org/10.1093/acprof:oso/9780199535255.001.0001.

[15] T. M. Chan and R. Williams, *Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky*, in Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, R. Krauthgamer, ed., SIAM, 2016, pp. 1246–1255, https://doi.org/10.1137/1.9781611974331.ch87.

[16] Y. Chen, M. Grohe, and B. Lin, *The hardness of embedding grids and walls*, in Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers, H. L. Bodlaender and G. J. Woeginger, eds., vol. 10520 of Lecture Notes in Computer Science, Springer, 2017, pp. 180–192, https://doi.org/10.1007/978-3-319-68705-6_14.

[17] H. Dell and J. 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, I. Diakonikolas, D. Kempe, and M. Henzinger, eds., ACM, 2018, pp. 281–288, https://doi.org/10.1145/3188745.3188920.

[18] H. Dell, J. Lapinskas, and K. Meeks, *Approximately counting and sampling small witnesses using a colourful decision oracle*, in Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020, S. Chawla, ed., SIAM, 2020, pp. 2201–2211, https://doi.org/10.1137/1.9781611975994.135.

[19] H. Dell, M. Roth, and P. Wellnitz, *Counting answers to existential questions*, in 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece., C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi, eds., vol. 132 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019, pp. 113:1–113:15, https://doi.org/10.4230/LIPIcs.ICALP.2019.113.

[20] J. Díaz, M. J. Serna, and D. M. Thilikos, *Counting H-colorings of partial k-trees*, Theor. Comput. Sci., 281 (2002), pp. 291–309, https://doi.org/10.1016/S0304-3975(02)00017-8.

[21] M. Dyer, L. A. Goldberg, C. Greenhill, and M. Jerrum, *The relative complexity of approximate counting problems*, Algorithmica, 38 (2004), pp. 471–500, https://doi.org/10.1007/s00453-003-1073-y.

[22] M. E. Dyer, L. A. Goldberg, and M. Jerrum, *An approximation trichotomy for boolean #csp*, J. Comput. Syst. Sci., 76 (2010), pp. 267–277, https://doi.org/10.1016/j.jcss.2009.08.003, https://doi.org/10.1016/j.jcss.2009.08.003.

[23] H.-D. Ebbinghaus, J. Flum, and W. Thomas, *Mathematical Logic*, Undergraduate Texts in Mathematics, Springer-Verlag New York, 1994.

[24] M. R. Fellows, G. Fertin, D. Hermelin, and S. 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, L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, eds., vol. 4596 of Lecture Notes in Computer Science, Springer, 2007, pp. 340–351, https://doi.org/10.1007/978-3-540-73420-8_31.

[25] I. Florescu, *Probability and Stochastic Processes*, Wiley-Blackwell, 2014.
[26] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer, 2006.
[27] J. Focke and M. Roth, *Counting small induced subgraphs with hereditary properties*, CoRR, abs/2111.02277 (2021).
[28] A. Gajentaan and M. H. Overmars, *On a class of $O(n^2)$ problems in computational geometry*, Comput. Geom., 45 (2012), pp. 140–152, https://doi.org/10.1016/j.comgeo.2011.11.006.
[29] A. Galanis, L. A. Goldberg, and M. Jerrum, *A complexity trichotomy for approximately counting list* H*-colorings*, ACM Trans. Comput. Theory, 9 (2017), pp. 9:1–9:22, https://doi.org/10.1145/3037381.
[30] J. Gao, R. Impagliazzo, A. Kolokolova, and R. Williams, *Completeness for first-order properties on sparse structures with algorithmic applications*, ACM Trans. Algorithms, 15 (2019), pp. 23:1–23:35, https://doi.org/10.1145/3196275.
[31] S. Guillemot and F. Sikora, *Finding and counting vertex-colored subtrees*, Algorithmica, 65 (2013), pp. 828–844, https://doi.org/10.1007/s00453-011-9600-8.
[32] H. Guo, C. Liao, P. Lu, and C. 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, I. Diakonikolas, D. Kempe, and M. Henzinger, eds., ACM, 2018, pp. 926–939, https://doi.org/10.1145/3188745.3188934.
[33] H. Guo, C. Liao, P. Lu, and C. Zhang, *Zeros of holant problems: Locations and algorithms*, ACM Trans. Algorithms, 17 (2021), pp. 4:1–4:25, https://doi.org/10.1145/3418056.
[34] S. Janson, T. Łuczak, and A. Rucinski, *Random Graphs*, John Wiley & Sons, 2000, https://doi.org/10.1002/9781118032718.
[35] M. Jerrum and K. Meeks, *The parameterised complexity of counting connected subgraphs and graph motifs*, Journal of Computer and System Sciences, 81 (2015), pp. 702 – 716, https://doi.org/10.1016/j.jcss.2014.11.015.
[36] M. Jerrum, A. Sinclair, and E. Vigoda, *A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries*, J. ACM, 51 (2004), pp. 671–697, https://doi.org/10.1145/1008731.1008738.
[37] M. Jerrum, L. G. Valiant, and V. V. Vazirani, *Random generation of combinatorial structures from a uniform distribution*, Theor. Comput. Sci., 43 (1986), pp. 169–188, https://doi.org/10.1016/0304-3975(86)90174-X, https://doi.org/10.1016/0304-3975(86)90174-X.
[38] D. M. Kane, S. Lovett, and S. Moran, *Near-optimal linear decision trees for k-SUM and related problems*, J. ACM, 66 (2019), pp. 16:1–16:18, https://doi.org/10.1145/3285953.
[39] I. Koutis, *Constrained multilinear detection for faster functional motif discovery*, Inf. Process. Lett., 112 (2012), pp. 889–892, https://doi.org/10.1016/j.ipl.2012.08.008.
[40] V. Lacroix, C. G. Fernandes, and M.-F. Sagot, *Motif search in graphs: Application to metabolic networks*, IEEE/ACM Trans. Comput. Biol. Bioinformatics, 3 (2006), pp. 360–368, https://doi.org/10.1109/TCBB.2006.55, https://doi.org/10.1109/TCBB.2006.55.
[41] L. Lovász, *Large Networks and Graph Limits*, vol. 60 of Colloquium Publications, American Mathematical Society, 2012, https://web.cs.elte.hu/~lovasz/bookxx/hombook-almost.final.pdf.
[42] D. Marx, *Can you beat treewidth?*, Theory of Computing, 6 (2010), pp. 85–112, https://doi.org/10.4086/toc.2010.v006a005.
[43] K. Meeks, *The challenges of unbounded treewidth in parameterised subgraph counting problems*, Discrete Applied Mathematics, 198 (2016), pp. 170 – 194, https://doi.org/10.1016/j.dam.2015.06.019.
[44] K. Meeks, *Randomised enumeration of small witnesses using a decision oracle*, Algorithmica, 81 (2019), pp. 519–540, https://doi.org/10.1007/s00453-018-0404-y.
[45] M. Müller, *Randomized approximations of parameterized counting problems*, in Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings, H. L. Bodlaender and M. A. Langston, eds., vol. 4169 of Lecture Notes in Computer Science, Springer, 2006, pp. 50–59, https://doi.org/10.1007/11847250_5.
[46] M. Patrascu and R. Williams, *On the possibility of faster SAT algorithms*, in Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010, M. Charikar, ed., SIAM, 2010, pp. 1065–1075, https://doi.org/10.1137/1.9781611973075.86.
[47] L. G. Valiant, *The complexity of computing the permanent*, Theor. Comput. Sci., 8 (1979), pp. 189–201, https://doi.org/10.1016/0304-3975(79)90044-6.
[48] L. G. Valiant and V. V. Vazirani, *NP is as easy as detecting unique solutions*, Theor.

Comput. Sci., 47 (1986), pp. 85–93, https://doi.org/10.1016/0304-3975(86)90135-0.

[49] R. 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, 2014, pp. 80:1–80:6, https://doi.org/10.1145/2603088.2603121.

[50] R. Williams, *Faster all-pairs shortest paths via circuit complexity*, SIAM J. Comput., 47 (2018), pp. 1965–1985, https://doi.org/10.1137/15M1024524.

[51] V. V. Williams, *Hardness of easy problems: Basing hardness on popular conjectures such as the Strong Exponential Time Hypothesis (invited talk)*, in 10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece, T. Husfeldt and I. A. Kanj, eds., vol. 43 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 17–29, https://doi.org/10.4230/LIPIcs.IPEC.2015.17.