Nearly Optimal Independence Oracle Algorithms for Edge Estimation in Hypergraphs

Holger Dell
Goethe University Frankfurt, Germany
IT University of Copenhagen and Basic Algorithms Research Copenhagen (BARC), Denmark

John Lapinskas
University of Bristol, UK

Kitty Meeks
University of Glasgow, UK

Abstract

Consider a query model of computation in which an n-vertex k-hypergraph can be accessed only via its independence oracle or via its colourful independence oracle, and each oracle query may incur a cost depending on the size of the query. Several recent results (Dell and Lapinskas, STOC 2018; Dell, Lapinskas, and Meeks, SODA 2020) give efficient algorithms to approximately count the hypergraph's edges in the colourful setting. These algorithms immediately imply fine-grained reductions from approximate counting to decision, with overhead only $\log^{\Theta(k)} n$ over the running time n^α of the original decision algorithm, for many well-studied problems including k-Orthogonal Vectors, k-SUM, subgraph isomorphism problems including k-Clique and colourful-H, graph motifs, and k-variable first-order model checking.

We explore the limits of what is achievable in this setting, obtaining unconditional lower bounds on the oracle cost of algorithms to approximately count the hypergraph's edges in both the colourful and uncoloured settings. In both settings, we also obtain algorithms which essentially match these lower bounds; in the colourful setting, this requires significant changes to the algorithm of Dell, Lapinskas, and Meeks (SODA 2020) and reduces the total overhead to $\log^{\Theta(k-\alpha)} n$. Our lower bound for the uncoloured setting shows that there is no fine-grained reduction from approximate counting to the corresponding uncoloured decision problem (except in the case $\alpha \geq k-1$): without an algorithm for the colourful decision problem, we cannot hope to avoid the much larger overhead of roughly $n^{(k-\alpha)^2/4}$. The uncoloured setting has previously been studied for the special case $k = 2$ (Peled, Ramamoorthy, Rashtchian, Sinha, ITCS 2018; Chen, Levi, and Waingarten, SODA 2020), and our work generalises the existing algorithms and lower bounds for this special case to $k > 2$ and to oracles with cost.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Oracles and decision trees; Mathematics of computing → Graph algorithms

Keywords and phrases Graph oracles, Fine-grained complexity, Approximate counting, Hypergraphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.54

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2211.03874` [14]

Funding *Kitty Meeks*: Supported by EPSRC grant EP/V032305/1.

1 Introduction

Many decision problems in computer science, particularly those in NP, can naturally be expressed in terms of determining the existence of a witness. For example, solving SAT requires determining the existence of a satisfying assignment to a CNF formula. All such problems Π naturally give rise to a counting version #Π, in which we ask for the number of witnesses. It is well-known that #Π is often significantly harder than Π; for example, Toda's theorem implies that it is impossible to solve #P-complete counting problems in polynomial time with access to an NP-oracle unless the polynomial hierarchy collapses. However, the same is not true for *approximately* counting witnesses (to within a factor of two, say). For example, it is known that: if Π is a problem in NP, then there is an FPRAS for #Π using an NP-oracle [30]; if Π is a problem in $W[i]$, then there is an FPTRAS for #Π using a $W[i]$-oracle [25]; and that the Exponential Time Hypothesis is equivalent to the statement that there is no subexponential-time approximation algorithm for #3-SAT [12].

In this paper we are concerned with analogous results in the fine-grained setting, which considers exact running times rather than coarse-grained classifications such as polynomial, FPT, or subexponential; such results turn out to be inextricably bound to graph oracle results of independent interest.

Past work in this area has focused on the family of *uniform witness problems* [13]. Roughly speaking, these are problems which can be expressed as counting edges in a k-hypergraph G in which the edges correspond to witnesses and induced subgraphs correspond to sub-problems. (See Section 3 for a detailed definition.) Many of the most important problems in fine-grained and parameterised complexity can be expressed as uniform witness problems including k-SUM, k-OV, k-Clique, Hamming weight-k solutions to CNFs, Size-k Graph Motif, most subgraph detection problems (including weighted problems such as Zero-Weight k-Clique and Negative-Weight Triangle), and first-order model-checking [13], in addition to certain database queries [16] and patterns in graphs [9]. Here k may be either a constant, as in the case of k-SUM, or a parameter, as in the case of k-Clique. In this setting, invoking a decision algorithm on a sub-problem of the original problem corresponds to invoking an oracle to test, given a set of vertices S, whether the induced subgraph $G[S]$ contains any edges; this oracle is called an *independence oracle* for G and is well-studied in its own right (see Section 4 for an overview).

Surprisingly, there is a partial analogue of the above reductions from approximate counting to decision in this setting. If the vertices of G are coloured, given a set $S \subseteq V(G)$, a *colourful independence oracle* tests whether $G[S]$ contains any edges with one vertex of each colour. This typically corresponds to a natural colourful variant of the original decision problem – for example, for k-Clique, it corresponds to deciding whether a k-coloured graph contains a size-k clique with one vertex of each colour. These oracles are again well-studied in their own right (see Section 4), and for many but not all uniform witness problems they can be efficiently simulated using the independence oracle. Given access to a colourful independence oracle for a graph G, we can count G's edges to within a factor of $1 \pm \varepsilon$ using $\varepsilon^{-2}k^{\mathcal{O}(k)}\log^{\Theta(k)} n$ oracle queries [13]. (See [4] for an improvement to the log factor.) In fact, we can say more – if we can simulate the colourful independence oracle in time n^{α_k} with $\alpha_k \geq 1$, then these queries dominate the running time and we obtain an approximate counting algorithm with running time $n^{\alpha_k} \cdot \varepsilon^{-2}k^{\mathcal{O}(k)}\log^{\Theta(k)} n$ in the usual word-RAM model. Translating back out of the oracle setting, this means that if we simulate the oracle by running an algorithm for the colourful decision problem, then for constant k and ε, we obtain an approximate counting algorithm with only polylogarithmic overhead over that decision algorithm. This result has led to several improved approximate counting algorithms – see [13] for applications to k-OV over finite fields and graph motifs, [16] for applications to database queries, and [9] for applications to patterns in graphs.

We are left with two major open problems of concern to researchers in fine-grained complexity, parameterised complexity and graph oracles, and we expect our paper to be of interest to all three communities. First, can the result of [13] be generalised from colourful independence oracles to independence oracles? This would imply, for example, a fine-grained reduction from approximate induced sub-hypergraph counting to induced sub-hypergraph detection. In this setting, efficiently simulating the colourful independence oracle using the independence oracle requires solving a long-standing open problem – see Section 3 – so the result of [13] does not straightforwardly apply. Second, in the parameterised setting, the factor of $\log^{\Theta(k)} n$ is not truly polylogarithmic, but equivalent to a factor of $k^{\mathcal{O}(k)}n^{o(1)}$. Can it be improved to $\log^{\mathcal{O}(1)} n$?

In this paper, we answer both questions, and in the process substantially generalise recent graph oracle results for the $k = 2$ case [11]. In both the colourful and uncoloured settings, we pin down the optimal oracle algorithm almost exactly. In both cases this algorithm improves on the current state of the art, and it allows for the desired fine-grained reductions if and only if the cost of calling the oracle on an x-vertex set (corresponding to the run-time of a decision algorithm on an x-element instance) is close to x^k. Moreover, our lower bounds are unconditional – they do not rely on conjectures such as SETH or FPT $\neq$ W[1].

In a little more detail, suppose for the moment that $\varepsilon = 1/2$, and that the cost of calling the oracle on an x-vertex set is x^{α_k} for some $\alpha_k \in [0, k]$. In the uncoloured setting, we define a function $g(k, \alpha_k) \approx (k - \alpha)^2/(4k)$ (see (2.1.1)) and show that an overhead of $2^{\mathcal{O}(k)}n^{g(k,\alpha_k)\pm o(1)}$ is both achievable and required; we have $g(k, \alpha_k) = 0$ when $\alpha_k \geq k - 1$, so in this regime we obtain a fine-grained reduction. In the colourful setting, we show that the $\log^{\Theta(k)} n$ overhead of [13, 4] can be improved to $\log^{\Theta(k-\alpha_k)} n$, but no further; thus polylogarithmic overhead is possible if and only if $k - \alpha_k \in \mathcal{O}(1)$ as $k \to \infty$. For general values of ε, both of our upper bounds have an additional multiplicative overhead of $\mathcal{O}(\varepsilon^{-2})$, which is common in approximate counting algorithms.

In the rest of the paper, we state our results for graph oracles more formally in Section 2, followed by their (immediate) corollaries for uniform witness problems in Section 3. We then give an overview of related work in Section 4, followed by a brief description of our proof techniques in Section 5 and a generalisation to non-polynomial running times in Section 6. A full version of the paper is available as [14].

2 Oracle results

Our results are focused on two graph oracle models on k-hypergraphs: independence oracles and colourful independence oracles. Both oracles are well-studied in their own right from a theoretical perspective, as they are both natural generalisations of group testing from unary relations to k-ary relations, and the apparent separation between them in power is already a source of substantial interest. They also provide a point of comparison for a rich history of sublinear-time algorithms for oracles which provide more local information, such as degree oracles. See the introduction of [11] for a more detailed overview of the full motivation, and Section 4 for a survey of past results.

In both the colourful and uncoloured case, while formally the oracles are bitstrings and a query takes $\mathcal{O}(1)$ time, in order to obtain reductions from approximate counting problems to decision problems in Section 3 we will simulate oracle queries using a decision algorithm. As such, rather than focusing on the *number* of queries as a computational resource, we define a more general *cost function* which will correspond to the running time of the algorithm used to simulate the query; thus the cost of a query will scale with its size. In our application, this allows for more efficient reductions by exploiting cheap queries, while also substantially strengthening our lower bounds. Indeed, simulating an oracle query typically requires between poly(k) and poly(n) time, so a lower bound on the total number of queries required would tell us very little; meanwhile, setting the cost of all queries to 1 in our results yields tight bounds for the number of queries required.

We are also concerned with the *running times* of our oracle algorithms, again due to our applications in Section 3. We work in the standard RAM-model of computation with $\Theta(\log n)$ bits per word and access to the usual $\mathcal{O}(1)$-time arithmetic and logical operations on these words; in addition, oracle algorithms can perform oracle queries, which are considered to take $\mathcal{O}(1)$ time.

As shorthand, for all real $x, y > 0$ and $\varepsilon \in (0,1)$, we say that x is an *ε-approximation* to y if $|x - y| < \varepsilon y$. We define an *ε-approximate counting algorithm* to be an oracle algorithm that is given n and k as explicit input, is given access to an oracle representing an n-vertex k-hypergraph G, and outputs an ε-approximation to the number of edges of G, denoted by $e(G)$. We allow ε to be part of the input (for upper bounds) or fixed (for lower bounds).

2.1 Our results for the uncoloured independence oracle

Given a k-hypergraph G with vertex set $[n]$, the *(uncoloured) independence oracle* is the bitstring $\texttt{IND}(G)$ such that for all sets $S \subseteq [n]$, $\texttt{IND}(G)_S = 1$ if $G[S]$ contains no edges and 0 otherwise. Thus a query to $\texttt{IND}(G)_S$ allows us to test whether or not the induced subgraph $G[S]$ contains an edge. We define the *cost* of an oracle call $\texttt{IND}(G)_S$ to be a polynomial function of the form $\mathrm{cost}_k(S) = |S|^{\alpha_k}$, where the map $k \mapsto \alpha_k$ satisfies $\alpha_k \in [0,k]$ but is otherwise arbitrary. (This upper bound is motivated by the fact that we can trivially enumerate all edges of G by using $\mathcal{O}(n^k)$ queries to all size-k subsets of $[n]$, incurring oracle cost at most $n^k \cdot k^{\alpha_k}$.)

It is not too hard to show that the naive $\mathcal{O}(n^k)$-cost exact edge-counting algorithm of querying every possible edge and the naive $\mathcal{O}(n^{\alpha_k})$-cost algorithm to decide whether any edge is present by querying $[n]$ are both essentially optimal. For approximate counting we prove the following, where for all real numbers x we write $\lfloor x \rceil := \lfloor x + 1/2 \rfloor$ for the value of x rounded to the nearest integer, rounding up in case of a tie.

▶ **Theorem 1** (Uncoloured independence oracle, polynomial cost function). *Let $\alpha_k \in [0,k]$ for all $k \geq 2$, let $\mathrm{cost}_k(x) = x^{\alpha_k}$, and let*

$$g(k,\beta) := \frac{1}{k} \cdot \left\lfloor \frac{k-\beta}{2} \right\rceil \cdot \left(k - \beta - \left\lfloor \frac{k-\beta}{2} \right\rceil \right). \tag{2.1.1}$$

There is a randomised ε-approximate counting algorithm $\texttt{Uncol}(\texttt{IND}(G), \varepsilon, \delta)$ with failure probability at most δ, worst-case running time

$$\mathcal{O}\Big(\log(1/\delta)\big(k^{5k} + \varepsilon^{-2} 2^{5k} \log^5 n \cdot n^{g(k,1)} \cdot n\big)\Big),$$

and worst-case oracle cost

$$\mathcal{O}\Big(\log(1/\delta)\big(k^{7k} + \varepsilon^{-2} 2^{5k} \log^5 n \cdot n^{g(k,\alpha_k)} \cdot n^{\alpha_k}\big)\Big)$$

under cost_k. Moreover, every randomised $(1/2)$-approximate edge-counting $\texttt{IND}$-oracle algorithm with failure probability at most $1/10$ has $\Omega((n^{g(k,\alpha_k)}/k^{3k}) \cdot n^{\alpha_k})$ worst-case expected oracle cost under cost_k.

Observe that the polynomial overhead $n^{g(k,\alpha_k)}$ of approximate counting over decision is roughly equal to $n^{(k-\alpha_k)^2/(4k)}$. If $\alpha_k = 0$, then the worst-case oracle cost of an algorithm is simply the worst-case number of queries that it makes. Thus Theorem 1 generalises known matching upper and lower bounds of $\widetilde{\Theta}(\sqrt{n})$ queries in the graph case [11], both by allowing $k > 2$ and by allowing $\alpha_k > 0$. (See Section 4 for more details.) Moreover, if $\alpha_k \geq k - 1$, then $g(k, \alpha_k) = 0$; thus in this case, Theorem 1 shows that approximate counting requires the same oracle cost as decision, up to a polylogarithmic factor. Taking $k = 2$ and $\alpha_k = 1$, this implies that whenever we can simulate an edge-detection oracle for a graph in linear time, then we can also obtain a linear-time approximate edge-counting algorithm (up to polylogarithmic factors). Analogous upper bounds on the running time and oracle cost of `Uncol` also hold for any "reasonable" cost function of the form $\mathrm{cost}_k(n) = n^{\alpha_k + o(1)}$; for details, see Section 6 and Theorem 10.

2.2 Our results for the colourful independence oracle

Given a k-hypergraph G with vertex set $[n]$, the *colourful independence oracle* is the bitstring $\mathtt{cIND}(G)$ such that for all disjoint sets $S_1, \dots, S_k \subseteq [n]$, $\mathtt{cIND}(G)_{S_1,\dots,S_k} = 1$ if G contains no edge $e \in E(G)$ with $|S_i \cap e| = 1$ for all i, and 0 otherwise. We view $S_1, \dots, S_k$ as colour classes in a partial colouring of $[n]$; thus a query to $\mathtt{cIND}(G)_{S_1,\dots,S_k}$ allows us to test whether or not G contains an edge with one vertex of each colour. (Note that we do not require $S_1 \cup \dots \cup S_k = [n]$.) Analogously to the uncoloured case, we define the *cost* of an oracle call $\mathtt{cIND}(G)_{S_1,\dots,S_k}$ to be a polynomial function of the form $\mathrm{cost}_k(S_1, \dots, S_k) = \mathrm{cost}_k(|S_1| + \dots + |S_k|) = (|S_1| + \dots + |S_k|)^{\alpha_k}$, where the map $k \mapsto \alpha_k$ satisfies $\alpha_k \in [0, k]$ but is otherwise arbitrary.

It is not too hard to show that the naive $\mathcal{O}(n^k)$-cost exact edge-counting algorithm of querying every possible edge and the naive $\mathcal{O}((k^k/k!)n^{\alpha_k})$-cost algorithm to decide whether any edge is present by randomly colouring the vertices are both essentially optimal, and indeed we prove as Proposition 65 of the full version that any such decision algorithm requires cost $\Omega(n^{\alpha_k})$. For approximate counting, we prove the following.

▶ **Theorem 2** (Colourful independence oracle, polynomial cost function). *Let $\alpha_k \in [0, k]$ for all $k \geq 2$, let $\mathrm{cost}_k(x) = x^{\alpha_k}$, and let $T := \log(1/\delta)\varepsilon^{-2}k^{27k}\log^{4(k-\lceil\alpha_k\rceil)+18} n$. There is a randomised ε-approximate edge counting algorithm $\mathtt{Count}(\mathtt{cIND}(G), \varepsilon, \delta)$ with worst-case running time $\mathcal{O}(T \cdot n^{\max(1,\alpha_k)})$, worst-case oracle cost $\mathcal{O}(T \cdot n^{\alpha_k})$ under cost_k, and failure probability at most δ. Moreover, every randomised $(1/2)$-approximate edge counting $\mathtt{cIND}$-oracle algorithm with failure probability at most $1/10$ has worst-case oracle cost*

$$\Omega\left(k^{-9k}\left(\frac{\log n}{\log\log n}\right)^{k-\lfloor\alpha_k\rfloor-3} \cdot n^{\alpha_k}\right)$$

under cost_k.

The upper bound replaces a $\log^{\Theta(k)} n$ term in the query count of the previous best-known algorithm ([5] for $\alpha_k = 0$) by a $\log^{\Theta(k-\alpha_k)} n$ term in the multiplicative overhead over decision, giving polylogarithmic overhead over decision when $k - \alpha_k = \mathcal{O}(1)$. The lower bound shows that this term is necessary and cannot be reduced to $\log^{\mathcal{O}(1)} n$; this is a new result even for $\alpha_k = 0$. (See Section 4 for more details.) Analogous upper bounds on the running time and oracle cost of `Count` also hold for any "reasonable" cost function of the form $\mathrm{cost}_k(n) = n^{\alpha_k + o(1)}$; for details, see Section 6 and Theorem 10.

2.3 Approximate sampling results

There is a known fine-grained reduction from approximate sampling to approximate counting [13]. Strictly speaking this, reduction is proved for $\alpha_k = 1$ with a colourful independence oracle, but the only actual use of the oracle in the reduction is to enumerate all edges in a set X with $\mathcal{O}(|X|^k)$ size-k queries, so it transfers immediately to our setting. The upper bounds of Theorems 1 and 2 therefore also yield approximate sampling algorithms with overhead $2^{\mathcal{O}(k)} \log^{\mathcal{O}(1)} n$ over approximate counting.

2.4 A parameterised complexity motivation for our lower bound results

To understand an important motivation for the lower bounds in our results, consider as an example the longest path problem: Given (G, k), does there exist a simple path of length k? A long sequence of works in parameterised complexity led to a spectacular algorithm [7] for this problem in undirected graphs that runs in time $1.66^k \cdot \mathrm{poly}(n)$. There is a somewhat shorter sequence of works for the corresponding approximate counting version of the problem, which culminated in a $4^k \,\mathrm{poly}(n)$-time algorithm [8, 20].

Instead of designing ever-more sophisticated algorithms for approximately counting k-paths in order to get closer to the running time of the decision problem, our dream result would instead be a subexponential-time approximate-counting-to-decision reduction that uses the decision problem in a black-box fashion and causes only a factor $2^{o(k)} \,\mathrm{poly}(n)$ overhead in the running time. This way, any improvements to the decision algorithm would automatically carry over. One way to formalise what the black-box can do is captured by defining the k-hypergraph whose edges are the k-paths of the underlying graphs; using an algorithm for the k-path decision problem, it is trivial to simulate the independence oracle and easy to simulate the colourful independence oracle for this hypergraph.

Theorem 2 implies that any decision algorithm for k-path can be turned into an approximate counting algorithm by paying a $\log^{\mathcal{O}(k)} n$-factor overhead in the running time. While this is still a fixed-parameter tractable running time, it leads to a useless algorithm, since the running time is much worse than $c^k \,\mathrm{poly}(n)$. The main consequence of Theorem 2 for k-path stems not from this meaningless upper bound, but from the lower bound, which is new even for $\alpha_k = 0$: Our results imply that if the decision algorithm for k-path is formalized using the colourful independence oracle, then the overhead of the approximate-counting-to-decision reduction must be $\log^{\Omega(k)} n$, and so a subexponential-time reduction cannot exist. Conversely, if a useful approximate-counting-to-decision reduction exists, it cannot merely be based on the hypergraph whose edges consist of all k-paths; instead, the reduction would have to have access to and exploit the underlying structure of the original graph. We believe that this is a useful insight for the design of future algorithms for approximate counting.

3 Reductions from approximate counting to decision

Theorems 1 and 2 can easily be applied to obtain reductions from approximate counting to decision for many important problems in fine-grained and parameterised complexity. The following definition is taken from [13]; recall that a counting problem is a function $\#\Pi\colon \{0,1\}^* \to \mathbb{N}$ and its corresponding decision problem is defined via $\Pi = \{x \in \{0,1\}^* : \#\Pi(x) > 0\}$.

▶ **Definition 3.** *The decision problem* Π *is a* uniform witness problem *if there is a function that maps instances $x \in \{0,1\}^*$ to uniform hypergraphs G_x such that the following statements hold:*

(i) *$\#\Pi(x)$ is equal to the number $e(G_x)$ of edges in G_x;*
(ii) *$V(G_x)$ and the size $k(G_x)$ 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 $I_x(S) \in \{0,1\}^*$ such that $G_{I_x(S)} = G_x[S]$ and $|I_x(S)| \in \mathcal{O}(|x|)$.*

The set $E(G_x)$ is the set of witnesses *of the instance x.*

Intuitively, we can think of a uniform witness problem as a problem of counting witnesses in an instance x that can be naturally expressed as edges in a hypergraph G_x, in such a way that induced subgraphs of G_x correspond to sub-instances of x. This allows us to simulate a query to $\mathtt{IND}(G_x)_S$ by running a decision algorithm for Π on the instance $I_x(S)$, and if our decision algorithm runs on an instance y in time $T(|y|)$ then this simulation will require time $\widetilde{\mathcal{O}}(T(|S|))$. Typically there is only one natural map $x \mapsto G_x$, and so we consider it to be a part of the problem statement. Simulating the independence oracle in this way, the statement of Theorem 1 yields the following.

▶ **Theorem 4.** *Suppose $\alpha_k \in [1, k]$ for all $k \geq 2$. Let Π be a uniform witness problem. Suppose that given an instance x of Π, writing $n = |V(G_x)|$ and $k = k(G_x)$, there is an algorithm to solve Π on x with error probability at most $1/3$ in time $\widetilde{\mathcal{O}}(n^{\alpha_k})$. Then there is an ε-approximation algorithm for $\#\Pi(x)$ with error probability at most $1/3$ and running time*

$$k^{\mathcal{O}(k)} + \varepsilon^{-2} n^{\alpha_k} \cdot 2^{\mathcal{O}(k)} n^{g(k,\alpha_k)}\,.$$

Note that the running time of the algorithm for $\#\Pi$ is the sum of the oracle cost and the running time of the algorithm of Theorem 1; by requiring $\alpha_k \geq 1$, we ensure this is dominated by the oracle cost. (Indeed, for most uniform witness problems it is very easy to prove that every decision algorithm must read a constant proportion of the input, and so we will always have $\alpha_k \geq 1$.) The lower bound of Theorem 1 implies that the $n^{\alpha_k + g(k,\alpha_k)}$ term in the running time cannot be substantially improved with any argument that relativises; thus in simple terms, there is a generic fine-grained reduction from approximate counting to decision if and only if the decision algorithm runs in time $\Omega(n^{k-1})$.

It is instructive to consider an example. Take Π to be the problem Induced-H of deciding whether a given input graph G contains an induced copy of a fixed graph H. In this case, the hypergraph corresponding to an instance G will have vertex set $V(G)$ and edge set $\{X \subseteq V(G) \colon G[X] \simeq H\}$; thus the witnesses are vertex sets which induce copies of H in G. The requirements of Definition 3(i) and (ii) are immediately satisfied, and Definition 3(iii) is satisfied since deleting vertices from the hypergraph corresponds to deleting vertices of G. Thus writing $k = |V(H)|$, Theorem 4 gives us a reduction from approximate #Induced-H to Induced-H with overhead $\varepsilon^{-2} 2^{\mathcal{O}(k)} n^{g(k,\alpha_k)}$ over the cost of the decision algorithm. Moreover, on applying the fine-grained reduction from approximate sampling to counting in [13] we also obtain an approximate uniform sampling algorithm with overhead $\varepsilon^{-2} 2^{\mathcal{O}(k)} n^{g(k,\alpha_k)}$. Many more examples of uniform witness problems to which Theorem 4 applies can be found in the introduction of [13].

We now describe the corresponding result in the colourful oracle setting, which we now set out – again, the following definition is taken from [13].

▶ **Definition 5.** *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 $S_1, \dots, S_k$, deciding whether $\mathtt{cIND}_{G_x}(S_1, \dots, S_k) = 0$ holds.*

Continuing our previous example, in Colourful-Induced-H, we are given a (perhaps improper) vertex colouring of our input graph G, and we wish to decide whether G contains an induced copy of H with exactly one vertex from each colour. Simulating an oracle call to $\mathtt{cIND}(G_x)_{S_1,\dots,S_k}$ corresponds to running a decision algorithm for Colourful-Π on the instance $I_x(S_1 \cup \dots \cup S_k)$ with colour classes $S_1, \dots, S_k$, and if this decision algorithm runs on an instance y in time $T(|y|)$ then this simulation will require time $\widetilde{\mathcal{O}}(T(|S_1| + \dots + |S_k|))$. Simulating the colourful independence oracle in this way, the statement of Theorem 2 yields the following.

▶ **Theorem 6.** *Suppose $\alpha_k \in [1, k]$ for all $k \geq 2$. Let Π be a uniform witness problem. Suppose that given an instance x of Π, writing $n = |V(G_x)|$ and $k = k(G_x)$, there is an algorithm to solve* Colourful-Π *on x with error probability at most $1/3$ in time $\mathcal{O}(n^{\alpha_k})$. Then there is an ε-approximation algorithm for $\#\Pi(x)$ with error probability at most $1/3$ and running time $\varepsilon^{-2} n^{\alpha_k} \cdot k^{\mathcal{O}(k)} (\log n)^{\mathcal{O}(k-\alpha_k)}$.*

As in the uncoloured case, the requirement $\alpha_k \geq 1$ ensures that the dominant term in the running time is the time required to simulate the required oracle queries, and the lower bound of Theorem 2 implies that the $\log^{\mathcal{O}(k-\alpha_k)} n$ term in the running time cannot be substantially improved with any argument that relativises. Again writing $k = |V(H)|$, Theorem 6 gives us a reduction from approximate #Induced-H to Colourful-Induced-H with overhead $\varepsilon^{-2}k^{\mathcal{O}(k)}(\log n)^{\mathcal{O}(k-\alpha_k)}$ over the cost of the decision algorithm. This result improves on the reduction of [13, Theorem 1.7] by a factor of $\log^{\Omega(\alpha_k)} n$, and using the fine-grained reduction from approximate sampling to counting in [13] it can immediately be turned into an approximate uniform sampling algorithm.

Observe that in most cases, there is far less overhead over decision in applying Theorem 6 to reduce #Induced-H to Colourful-Induced-H than there is in applying Theorem 4 to reduce #Induced-H to Induced-H. In some cases, such as the case where H is a k-clique, there are simple fine-grained reductions from Colourful-Induced-H to Induced-H, and in this case Theorem 6 is an improvement. However, it is not known whether the same is true of all choices of H, and indeed even an FPT reduction from Colourful-Induced-H to Induced-H would imply the long-standing dichotomy conjecture for the embedding problem introduced in [18]. More generally, but still within the setting of uniform witness problems, the problem of detecting whether a graph contains a size-k set which either spans a clique or spans an independent set is in FPT by Ramsey's theorem, but its colourful version is W[1]-complete [22].

While the distinction between colourful problems and uncoloured problems is already well-studied in subgraph problems, these results strongly suggest that it is worth studying in many other contexts in fine-grained complexity as well. Indeed, it is easy to simulate `IND`(G) from `cIND`(G) with random colouring; thus the lower bound of Theorem 1 and the upper bound of Theorem 2 imply that there is a fine-grained reduction from uncoloured approximate counting to colourful decision, but not to uncoloured decision. By studying the relationship between colourful problems and their uncoloured counterparts, we may therefore hope to shed light on the relationship between approximate counting and decision.

Finally, we observe that the set of running times allowed by Theorems 4 and 6 may not be sufficiently fine-grained to derive meaningful results for some problems. In fine-grained complexity, even a subpolynomial improvement to a polynomial-time algorithm may be of significant interest – the classic example is the Negative-Weight-Triangle algorithm of [31], which runs in $n^3/e^{\Omega(\sqrt{\log n})}$ time on an n-vertex instance, compared to the naive $\mathcal{O}(n^3)$-time algorithm. In order to "lift" such improvements from decision problems to approximate counting, we must generalise the upper bounds of Theorems 1 and 2 to cost functions of the form $\mathrm{cost}_k(x) = x^{\alpha_k \pm o(1)}$ while maintaining low overhead. We do so in Theorem 10 for all "reasonable" cost functions, including any function of the form $\mathrm{cost}_k(x) = n^{\alpha_k}\log^{\beta_k} n$ and any function of the form $\mathrm{cost}_k(x) = n^{\alpha_k}e^{\pm(\log n)^{\gamma_k}}$ where $\gamma_k < 1$. A full list of technical requirements is given in Section 6, but the most important one is regular variation – this is a standard notion from probability theory for "almost polynomial" functions, and requiring it avoids pathological cases where (for example) we may have $\mathrm{cost}_k(x) = \mathcal{O}(x)$ as $x \to \infty$, but $\mathrm{cost}_k(2x_i) = \omega(\mathrm{cost}_k(x_i))$ as $i \to \infty$ along some sequence $(x_i \colon i \geq 1)$.

4 Discussion of related work

In order to compare algorithms without excessive re-definition of notation, throughout this subsection we consider the problem of ε-approximating the number of edges in an m-edge, n-vertex k-hypergraph.

Colourful and uncoloured independence oracles were introduced in [3] in the graph setting, then first generalised to hypergraphs in [6]. Edge estimation using these oracles was first studied in the graph setting (i.e. for $k = 2$) in [3], which gave an $\varepsilon^{-4}\log^{\mathcal{O}(1)} n$-query algorithm for colourful independence oracles and an $(\varepsilon^{-4} + \varepsilon^{-2}\min\{\sqrt{m}, n^2/m\})\log^{\mathcal{O}(1)} n = (\varepsilon^{-4} + \varepsilon^{-2}n^{2/3})\log^{\mathcal{O}(1)} n$-query algorithm for uncoloured independence oracles. The connection to approximate counting in fine-grained and parameterised complexity was first studied in [12].

For colourful independence oracles in the graph setting, [12] (independently from [3]) gave an algorithm using $\varepsilon^{-2}\log^{\mathcal{O}(1)} n$ `cIND` queries and $\varepsilon^{-2}n\log^{\mathcal{O}(1)} n$ adjacency queries. [1] subsequently gave a *non-adaptive* algorithm using $\varepsilon^{-6}\log^{\mathcal{O}(1)} n$ `cIND` queries.

The case of edge estimation in k-hypergraphs (i.e. for arbitrary $k \geq 2$) was first considered independently by [13, 4]; [13] gave an algorithm using $\varepsilon^{-2}k^{\mathcal{O}(k)}\log^{4k+\mathcal{O}(1)} n$ queries, while [4] gave an $\varepsilon^{-4}k^{\mathcal{O}(k)}\log^{4k+\mathcal{O}(1)}$-query algorithm. [13] also introduced a reduction from approximate sampling to approximate counting in this setting (which also applies in the uncoloured setting) with overhead $k^{\mathcal{O}(k)}\log^{\mathcal{O}(1)} n$. [5] then improved the query count further to $\varepsilon^{-2}k^{\mathcal{O}(k)}\log^{3k+\mathcal{O}(1)} n$.

In this paper, we give an algorithm with total query cost $\varepsilon^{-2}k^{\mathcal{O}(k)}\log^{4(k-\alpha_k)+\mathcal{O}(1)} n$ under $\mathrm{cost}_k(x) = x^{\alpha_k}$, giving polylogarithmic overhead when $\alpha_k \approx k$. We also give a lower bound which shows that a $\log^{\Theta(k-\alpha_k)}$ term is necessary; no lower bounds were previously known even for $\alpha_k = 0$ (i.e. the case where the total query cost equals the number of queries).

For uncoloured independence oracles of graphs, [11] improved the algorithm of [3] to use

$$\varepsilon^{-\Theta(1)}\min\{\sqrt{m}, n/\sqrt{m}\}\,\mathrm{polylog}\, n = \varepsilon^{-\Theta(1)}\sqrt{n}\,\mathrm{polylog}\, n \tag{4.0.1}$$

queries and gave a matching lower bound. (It is difficult to tell the exact value of the $\Theta(1)$ term from the proof, but it is at least 9 – see the definition of N in the proof of Lemma 3.9 on p. 15.) It is worth noting that the bound in (4.0.1) is stated as a function of both n and m. We believe that our results can be stated in such a way as well, but we defer doing so to the journal version of this paper.

To the best of our knowledge, no results on uncoloured edge estimation for $\alpha_k > 0$ or $k > 2$ have previously appeared in the literature. However, we believe it would be easy to partially generalise the proof of [3] to this setting. Very roughly speaking, their argument works by running a naive sampling algorithm and a more subtle branch-and-bound approximation algorithm in parallel, with the sampling algorithm running quickly on dense graphs and the branch-and-bound algorithm running quickly on sparse graphs. The main obstacle to generalising this approach would be the branch-and-bound approximation algorithm; however, by replacing it with a slower branch-and-bound *enumeration* algorithm for k-hypergraphs such as [23], we believe we would obtain worst-case oracle cost $k^{\mathcal{O}(k)} + \varepsilon^{-2}2^{\mathcal{O}(k)}n^{\alpha_k+(k-\alpha_k)/2}$ under $\mathrm{cost}_k(x) = x^{\alpha_k}$; this technique is well-known in the literature and also appears in e.g. [29]. By comparison (see Figure 1), the algorithm of Theorem 1 achieves a much smaller worst-case oracle cost of $k^{\mathcal{O}(k)} + \varepsilon^{-2}2^{\mathcal{O}(k)}n^{\alpha_k+g(k,\alpha_k)}$, where $g(k,\alpha_k) \approx (k-\alpha_k)^2/(4k) < (k-\alpha_k)/2$ and where $g(k,\alpha_k) = 0$ for $\alpha_k \geq k-1$. This substantially improves on the algorithm implicit in [3], and indeed is optimal up to a factor of $\varepsilon^{\Theta(1)}k^{\Theta(k)}$. Also, our algorithm has a better dependence on ε compared with [11] when $k = 2$; however, we bound the cost only in terms of n and not in terms of m.

Although a full survey is beyond the scope of this paper, there are natural generalisations of (colourful and uncoloured) independence oracles [27], and edge estimation problems are studied for other oracle models including neighbourhood access [28]. Other types of oracle are also regularly applied to fine-grained complexity in other models, notably including cut oracles [24, 2]. Perhaps surprisingly, we were unable to find any previous examples in the

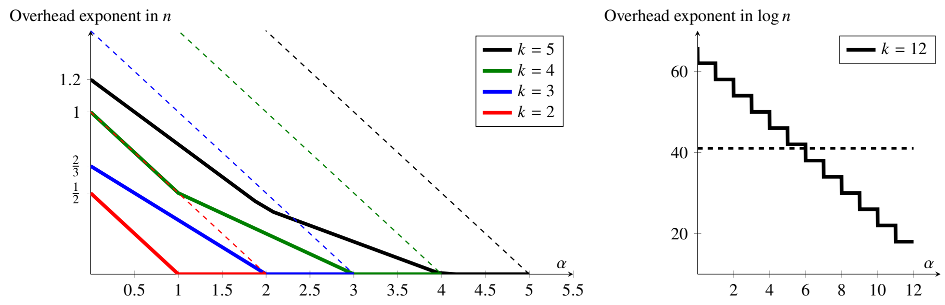

Figure 1 *Left:* If each IND-query of size x has cost x^{α_k}, then up to $k^{\mathcal{O}(k)} \log^{\mathcal{O}(1)} n$ factors, we show in Theorem 1 that $n^{g(k,\alpha_k)+\alpha_k}$ is the smallest possible IND-oracle cost to (1/2)-approximate the number of edges. Plotted here in *thick lines* is the overhead $\alpha \mapsto g(k, \alpha)$ in the exponent of n for $k \in \{2, 3, 4, 5\}$, and in *dashed lines* is the larger overhead exponent $\alpha \mapsto (k - \alpha)/2$ obtained from a naive generalisation of the techniques of [3]. *Right:* If each cIND-query of size x has cost x^{α_k}, then up to $k^{\mathcal{O}(k)}(\log n)^{\mathcal{O}(1)}(\log\log n)^{\mathcal{O}(k-\alpha_k)}$ factors, we show in Theorem 2 that $n^{\alpha_k} \log^{4(k-\lceil\alpha_k\rceil)+18} n$ is the smallest possible cIND-oracle cost to (1/2)-approximate the number of edges. Plotted in *thick lines* is the overhead $\alpha \mapsto 4(k - \lceil\alpha\rceil) + 18$ in the exponent of $\log n$ for $k = 12$, and the *dashed line* depicts the overhead $\alpha \mapsto 3k + 5$ obtained by using the bound on the number of queries by [5]; our bound is better if $\lceil\alpha_k\rceil \geq k/4 + \Theta(1)$.

literature of unconditional oracle lower bounds relative to a general query cost function, and so our definitions and methods are novel in that sense. Of course, many existing works prove unconditional lower bounds in terms of query count [28], or consider algorithmic construction of graph oracles with bounded query times [19], or provide oracle algorithms with fast running times in addition to low query counts [26]. In our setting, however, a lower bound in terms of query cost is absolutely necessary. Recall that for us, query cost is the running time of an algorithm for the decision problem we are reducing to, and the algorithmic results of Theorems 4 and 6 all rely on smaller queries running faster; thus to prove they are "best possible" in any meaningful sense, we absolutely require the formal notion of query cost set out in Section 2.

Outside the oracle setting, it was recently proved [21] that any decision algorithm built around the representative family technique of [17] can be turned into an approximate counting algorithm with substantially lower overheads in k than those of [13]. More recently, several important decision problems in the fine-grained setting with $k = 3$ have been discovered to be "equivalent" to their *exact* counting versions [10]. This work is not directly comparable to ours, as they work in a substantially stronger setting and use a correspondingly weaker notion of equivalence. Their equivalence is in the sense of equivalence of conjectures – for example, they prove that if there is an $O(n^{2-\varepsilon})$-time algorithm for 3-SUM, then there exists $0 < \varepsilon' < \varepsilon$ such that there is an $O(n^{2-\varepsilon'})$-time algorithm for #3-SUM. We stress that for exact counting problems as studied in [10], such equivalence results are genuinely deep and surprising. However, for the approximate counting problems we study, analogous results are typically quite easy to prove via the standard combination of sampling and branch-and-bound approaches discussed above, and so we instead study the stronger notion of fine-grained reductions from approximate counting to decision. Where no such reductions exist, we aim to nail down the exact value of ε'. (Recall also that there are uniform witness problems whose decision versions are in FPT but whose exact counting versions are W[1]-hard [22], so we cannot hope to extend our results to exact counting in this setting.)

5 Proof techniques

5.1 Colourful upper bound

We first discuss the proof of the upper bound of Theorem 2, our cIND-oracle algorithm for edge estimation using the colourful independence oracle of an n-vertex k-hypergraph G. In [13], it is implicitly proved that a cIND-oracle algorithm that computes a "coarse approximation" to $e(G)$, which is accurate up to multiplicative error b, can be bootstrapped into a full ε-approximation algorithm with overhead $\varepsilon^{-2}2^{\mathcal{O}(k)}\log^{\mathcal{O}(1)} n \cdot b^2$. (See Theorem 49 of the full version for details.) It therefore suffices to improve the coarse approximation algorithm of [13] from $k^{\mathcal{O}(k)}\log^{\Theta(k)} n$ queries and $k^{\mathcal{O}(k)}\log^{\Theta(k)} n$ multiplicative error to $k^{\mathcal{O}(k)}\log^{\Theta(k-\alpha_k)} n$ query cost and $k^{\mathcal{O}(k)}\log^{\Theta(k-\alpha_k)} n$ multiplicative error. Moreover, by a standard colour-coding argument, it suffices to make this improvement when G is k-partite with vertex classes $V_1, \dots, V_k$ known to the algorithm.

Oversimplifying a little, and assuming n is a power of two, the algorithm of [13] works by guessing a probability vector $(Q_1, \dots, Q_k) \in \{1, 1/2, 1/4, \dots, 1/n\}^k$. It then deletes vertices from $V_1, \dots, V_k$ independently at random to form sets $\mathcal{X}_1, \dots, \mathcal{X}_k$, so that for all $v \in V_j$ we have $\mathbb{P}(v \in \mathcal{X}_j) = Q_j$. After doing so, in expectation, $Q_1 Q_2 \dots Q_k$ proportion of the edges of G remain in the induced k-partite subgraph $G[\mathcal{X}_1, \dots, \mathcal{X}_k]$. If $e(G) \ll 1/(Q_1 \dots Q_k)$, it is easy to show with a union bound that no edges are likely to remain. What is more surprising is that there exist $q_1, \dots, q_k$ with $q_1 \dots q_k \approx 1/e(G)$ such that if $\vec{Q} = \vec{q}$, then at least one edge is likely to remain in $G[\mathcal{X}_1, \dots, \mathcal{X}_k]$. Thus the algorithm of [13] iterates over all $\log^{\Theta(k)} n$ possible values of $\vec{Q}$, querying $\texttt{cIND}(G)$ on $\mathcal{X}_1, \dots, \mathcal{X}_k$ for each, and then outputs the least value m such that $e(G[\mathcal{X}_1, \dots, \mathcal{X}_k]) > 0$ for some $Q_1, \dots, Q_k$ with $1/(Q_1 \dots Q_k) = m$.

Our algorithm improves on this idea as follows. First, [13] does not actually prove the existence of the vector $\vec{q}$ described above – it relies on a coupling between the different guesses of $\vec{Q}$. We require not only the existence of $\vec{q}$ but also a structural result which may be of independent interest. For all $I \subseteq [k]$ and all $\zeta \in (0, 1]$, we define an (I, ζ)-*core* to be an induced subgraph $H = G[Y_1, \dots, Y_k]$ of G such that:

(i) H contains at least $k^{-\mathcal{O}(k)}$ proportion of the edges of G.

(ii) For all $i \in I$, the set Y_i is very small, containing at most $2/\zeta$ vertices.

(iii) For all $i \notin I$, every vertex of Y_i is contained in at most ζ proportion of edges in H.

As an example, the most extreme core is the *rooted star*: It consists of some vertices $r_i \in Y_i$ for all $i \in I$ and *all* k-partite edges e with $e \supseteq \{\, r_i : i \in I \,\}$. We prove in Lemma 56 of the full version that, for all $\zeta \in (0, 1]$, there is some $I \subseteq [k]$ such that G contains an (I, ζ)-core H.

Suppose for the moment that we are given the value of I, but not $Y_1, \dots, Y_k$. By property (i), it would then suffice to approximate $e(H)$ using $k^{\mathcal{O}(k)}\log^{\mathcal{O}(k-\alpha_k)} n$ query cost. If $|I| \geq \alpha_k$, then we can adapt the idea of the algorithm of [13], but taking $Q_j = 1$ for all $i \notin I$ to use only $\log^{\mathcal{O}(k-\alpha_k)} n$ queries in total; intuitively, this is possible due to property (ii), which implies that this is the "correct" setting. We set this algorithm out as `CoarseLargeCore` in Section 4.1.3 of the full version.

If instead $|I| \leq \alpha_k$, then we will exploit the fact that query cost decreases polynomially with instance size by breaking H into smaller instances. For all $i \notin I$, we randomly delete vertices from V_i with a carefully-chosen probability p. Property (iii), together with a martingale bound (see Lemma 59 of the full version), guarantees that the number of edges in the resulting hypergraph G' will be concentrated around its expectation of $p^k e(G)$. If we had access to $Y_1, \dots, Y_k$, we could then intersect V_i with Y_i for all $i \in I$ to obtain a substantially smaller instance, whose edges we could count with cheap queries; we could then divide the result by p^k to obtain an estimate for $e(G)$. Unfortunately we do not have access to $Y_1, \dots, Y_k$, but we can still break G' into smaller sub-hypergraphs by applying colour-coding to the vertex sets V_i with $i \in I$, and as long as $|I| \le \alpha_k$ this still gives enough of a saving in query cost to prove the result. We set this algorithm out as `CoarseSmallCore` in Section 4.1.2 of the full version.

Now, we are not in fact given the value of I in the (I, ζ)-core of G. But both `CoarseLargeCore` and `CoarseSmallCore` fail gracefully if they are given an incorrect value of I, returning an underestimate of $e(G)$ rather than an overestimate. We can therefore simply iterate over all 2^k possible values of I, applying `CoarseLargeCore` or `CoarseSmallCore` as appropriate, and return the maximum resulting estimate of $e(G)$. This proves the result.

5.2 Colourful lower bound

We now discuss the proof of the lower bound of Theorem 2. Using the minimax principle, to prove the bound for randomised algorithms, it is enough to give a pair of random graphs G_1 and G_2 with $e(G_2) \gg e(G_1)$ and prove that no *deterministic* algorithm A can distinguish between G_1 and G_2 with constant probability and worst-case oracle cost as in the bound. We base these random graphs on the main bottleneck in the algorithm described in the previous section: the need to check all possible values of $\mathcal{Q}$ in a k-partite k-hypergraph with an (I, ζ)-core where $|I| \approx k - \alpha_k$.

We take G_1 to be an Erdős-Rényi k-partite k-hypergraph with edge density $p := t^{-(k-\lfloor \alpha_k \rfloor - 2)/2}$. We take the vertex classes $V_1, \dots, V_k$ of G_1 to have equal size t, so that $t = n/k$. We then define a random complete k-partite graph H as follows. We first define a random vector $\vec{\mathcal{Q}}$ of probabilities, then take binomially random subsets $\mathcal{X}_1, \dots, \mathcal{X}_{k-\lfloor \alpha_k \rfloor - 2}$ of $V_1, \dots, V_{k-\lfloor \alpha_k \rfloor - 2}$, with $\mathbb{P}(v \in \mathcal{X}_j) = \mathcal{Q}_j$ for all $v \in V_j$. For $j \ge k - \lfloor \alpha_k \rfloor - 1$, we take $\mathcal{X}_j \subseteq V_j$ to contain a single uniformly random vertex. We then define H to be the complete k-partite graph with vertex classes $\mathcal{X}_1, \dots, \mathcal{X}_k$, and form $G_2 = G_1 \cup H$ by adding the edges of H to G_1. We choose $\mathcal{Q}$ in such a way that $\mathcal{Q}_1 \cdot \ldots \cdot \mathcal{Q}_{k-\lfloor \alpha_k \rfloor - 2}$ is guaranteed to be a bit larger than $pt^{\lfloor \alpha_k \rfloor + 2}$, so that $\mathbb{E}(e(H)) \gg \mathbb{E}(e(G_1))$. Intuitively, this corresponds to the situation of a randomly planted (I, ζ)-core where $I = \{k - \lfloor \alpha_k \rfloor - 1, \dots, k\}$ – we will show that the algorithm A needs to essentially guess the value of $\mathcal{Q}$ using expensive queries.

To show that a low-cost deterministic algorithm A cannot distinguish G_1 from G_2, suppose for simplicity that A is *non-adaptive*, so that its future oracle queries cannot depend on the oracle's past responses. In this setting, it suffices to bound the probability of a fixed query $S = (S_1, \dots, S_k)$ distinguishing G_1 from G_2.

It is not hard to show that without loss of generality we can assume $S_i \subseteq V_i$ for all $i \in [k]$. If $|S_j| \ll t$ for some $j \ge k - \lfloor \alpha_k \rfloor - 1$, then with high probability S_j will not contain the single "root" vertex of $\mathcal{X}_j$, so $H[S_1, \dots, S_k]$ will contain no edges and S will not distinguish G_1 from G_2. With some effort (a simple union bound does not suffice), this idea allows us to essentially restrict our attention to large, expensive queries S. However, if $|S_1| \dots |S_k|$ is large, then with high probability $G_1[S_1, \dots, S_k]$ will contain an edge, so again S will not distinguish G_1 from $G_2 = G_1 \cup H$. With some more effort, this allows us to essentially restrict our attention to queries where for some possible value $\vec{q}$ of $\vec{\mathcal{Q}}$ we have $|S_j| \approx 1/q_j$ for all $j \le k - \lfloor \alpha \rfloor - 2$; we choose these possible values to be far enough apart that such a query is only likely to distinguish G_1 from G_2 if $\vec{\mathcal{Q}} = \vec{q}$. There are roughly $((\log n)/\log\log n)^{k-\lfloor \alpha_k \rfloor - 2}$ possible values of $\mathcal{Q}_j$, so the result follows.

Of course, in our setting A may be adaptive, and this breaks the argument above. Since the query A makes depends on the results of past queries, we cannot bound the probability of a fixed query distinguishing G_1 from G_2 in isolation – we must condition on the results of past queries. This is not a small technical point – it is equivalent to allowing A to be adaptive in the first place. The most damaging implication is that we could have a query $S = (S_1, \dots, S_k)$ with $|S_1| \dots |S_k|$ very large but such that $G_1[S_1, \dots, S_k]$ contains no edges, because most of the potential edges have already been exposed as not existing in past queries. We are able to deal with this while preserving the spirit of the argument above, by arguing based on the number of unexposed edges rather than $|S_1| \dots |S_k|$, but it requires significantly more effort and a great deal of care.

5.3 Uncoloured upper bound

We now discuss the proof of the upper bound of Theorem 1. We adapt a classic framework for approximate counting algorithms that originated in [30], and that was previously applied to edge counting in [12]. We first observe that by using an algorithm from [23], we can enumerate the edges in an n-vertex k-hypergraph G with $2^{\mathcal{O}(k)} \log^{\mathcal{O}(1)} n \cdot e(G)$ queries to an independence oracle. Suppose we form an induced subgraph G_i of G by deleting vertices independently at random, keeping each vertex with probability 2^{-i}; then in expectation, we have $e(G_i) = 2^{-ki}e(G)$. If $e(G_i)$ is small, and $e(G_i) \approx \mathbb{E}(e(G_i))$, then we can efficiently count the edges of G_i using [23] and then multiply by 2^{ki} to obtain an estimate of $e(G)$. We can then simply iterate over all i from 0 to $\log n$ and return an estimate based on the first i such that $e(G_i)$ is small enough for [23] to return a value quickly.

Of course in general we do not have $e(G_i) \approx \mathbb{E}(e(G_i))$! One issue arises if, for some $r \in [k-1]$, every edge of G contains a common size-r set R – a "root". Then with probability $1 - 2^{-ri}$, at least one vertex in R will be deleted and G_i will contain no edges. We address this issue in the simplest way possible: by taking more samples. Roughly speaking, suppose we are given i, and that we already know (based on the failure of previous values of i to return a result) that $e(G) > n^{k-r-1}$ for some $0 \leq r \leq k-1$. This implies that G cannot have any "roots" of size greater than r. Rather than taking a single random subgraph G_i, we take $t_i \approx 2^{ri}$ independent copies of G_i and sum their edge counts using [23]; thus if G does contain a size-r root, we are likely to include it in the vertex set of at least one sample. Writing Σ_i for the sum of their edge counts, we then return $\Sigma_i/(t_i 2^{-ik})$ if the enumeration succeeds.

The exact expression we use for t_i is more complicated than 2^{ri}, due to the possibility of multiple roots – see Section 3.2.2 of the full version for a more detailed discussion – but the idea is the same. The proof that Σ_i is concentrated around its expectation is an (admittedly somewhat involved) application of Chebyshev's inequality, in which the rooted "worst cases" correspond to terms in the variance of Σ_i. We consider it surprising and interesting that such a conceptually simple approach yields an optimal upper bound, and indeed gives us a strong hint as to how we should prove the lower bound of Theorem 1.

5.4 Uncoloured lower bound

We finally discuss the proof of the lower bound of Theorem 1. As in the colourful case, we apply the minimax principle, so we wish to define random k-hypergraphs G_1 and G_2 with $e(G_2) \gg e(G_1)$ which cannot be distinguished by a low-cost deterministic algorithm A.

Our construction of G_1 and G_2 is natural from the above discussion, and the special case with $k = 2$ and $\alpha_k = 0$ is very similar to the construction used in [11]. We take G_1 to be an Erdős-Rényi k-hypergraph with edge probability roughly $k!/n^r$. We choose a random collection of size-r sets in $V(G_1)$ to be "roots", with a large constant number of roots present in expectation, and we define a k-hypergraph H to include every possible edge containing at least one of these roots as a subset. We then define $G_2 := G_1 \cup H$.

Similarly to the colourful lower bound, in the non-adaptive setting, any fixed query S_i with $|S_i|$ large is likely to contain an edge of G_1, and any fixed query with $|S_i|$ small is unlikely to contain a root and therefore unlikely to contain any edges of H; in either case, the query does not distinguish G_1 from G_2. Also as in the colourful case, generalising this argument from the non-adaptive setting to the adaptive setting requires a significant amount of care and effort.

6 More general cost functions for upper bounds

In our previous theorem statements we focused on polynomial cost functions of the form $\mathrm{cost}_k(S) = |S|^{\alpha_k}$ for some $\alpha_k \in [0, k]$, and these functions are easy to work with. However, even subpolynomial improvements to an algorithm's running time can be of interest and by considering more general cost functions we can lift these improvements from decision algorithms to approximate counting algorithms. To take a well-known example, in the negative-weight triangle problem, we are given an n-vertex edge-weighted graph G and asked to determine whether it contains a triangle of negative total weight; this problem is equivalent to a set of other problems under subcubic reductions, including APSP [32]. The naive $\Theta(n^3)$-time algorithm can be improved by a subpolynomial factor of $e^{\Theta(\sqrt{\log n})}$ [31], but as stated in the introduction our result would not lift this improvement from decision to approximate counting – we would need to take $k = 3$ and $\mathrm{cost}(n) = n^3/e^{\Theta(\sqrt{\log n})}$. (For clarity, in this specific case the problem is equivalent to its colourful version and a fine-grained reduction is already known [12, 13].)

While we cannot hope to say anything meaningful in the algorithmic setting with a fully general cost function, our results do extend to all cost functions which might reasonably arise as running times. A natural first attempt to formalise what we mean by "reasonable" would be to consider cost functions of the form $\mathrm{cost}_k(n) = n^{\alpha_k + o(1)}$ as $n \to \infty$. Unfortunately, such cost functions can still have a pathological property which makes a fine-grained reduction almost impossible: the $o(1)$ term might vary wildly between different values of n. For example, if we take $\mathrm{cost}_k(n) = n^{\alpha_k + \sin(\pi n/2)/\sqrt{\log(n)}}$, then we have $\mathrm{cost}_k(n) = n^{\alpha_k + o(1)}$, but $\mathrm{cost}_k(2n)/\mathrm{cost}_k(n)$ could be $\omega(\mathrm{polylog}(n))$, $\Theta(1)$ or $o(1/\mathrm{polylog}(n))$ depending on whether n is congruent to 3, 2 or 1 modulo 4 (respectively). It is even possible to construct a similar example which is monotonically increasing. We will need to be able to approximate $f(Ax)$ by $A^{\alpha_k} f(x)$ and so we require something slightly stronger, borrowing a standard notion from the probability literature for distributions which are "almost power laws".

▶ **Definition 7.** *A function $f\colon (0,\infty) \to \mathbb{R}$ is* regularly-varying *if, for all fixed $A > 0$,*

$$\lim_{x\to\infty} |f(Ax)/f(x)| \in (0, \infty).$$

We say f is slowly-varying *if this limit is always* 1.

Observe that any cost function likely to arise as a running time is regularly-varying. While Definition 7 may look overly general for our purposes, in fact it gives us exactly what we need – regularly-varying functions are all of the form $f(x) = x^{\alpha + o(1)}$ as $x \to \infty$, are eventually increasing whenever $\alpha > 0$, and satisfy very strict bounds on $f(Ax)/f(x)$ as $x \to \infty$ as detailed in the following lemma. (These properties are all well-known, see e.g. Feller [15].) With these properties in hand, the extra generality of Definition 7 adds very little overhead to our existing arguments.

▶ **Lemma 8.** *Let $f\colon (0,\infty) \to \mathbb{R}$ be a function. Then f is regularly-varying if and only if there exists a unique $\alpha \in \mathbb{R}$, called the* index *of f, and a unique slowly-varying function $\sigma\colon (0,\infty) \to \mathbb{R}$, called the* slowly-varying component *of f, such that $f(x) = x^{\alpha}\sigma(x)$. Moreover, any regularly-varying function f and slowly-varying function σ satisfy the following properties.*

(i) *$\sigma(x) = x^{o(1)}$ as $x \to \infty$.*

(ii) *For all fixed $A > 0$, $\lim_{x\to\infty}(\sigma(Ax)/\sigma(x)) = 1$. Moreover, for all closed intervals $I \subseteq \mathbb{R}$, this limit is uniform over all $A \in I$.*

(iii) *For all $\delta > 0$, there exists x_0 such that for all $x \geq x_0$ and all $A_x \geq 1$,*

$$A_x^{-\delta} \leq \sigma(A_x x)/\sigma(x) \leq A_x^{\delta}.$$

(iv) *If f has positive index, then there exists x_0 such that f is strictly increasing on $[x_0,\infty)$.*

Our cost function will be a function of both the parameter k and the number n of vertices in the instance, and we will only require regular variation in n. We are now ready to state the technical requirements on our cost functions.

▶ **Definition 9.** *For each $k \geq 2$, let $\mathrm{cost}_k\colon (0,\infty) \to (0,\infty)$. We say that $\mathrm{cost} = \{\mathrm{cost}_k : k \geq 2\}$ is a* regularly-varying parameterised cost function *if $\mathrm{cost}_k(x) \leq x^k$ for all k and x, and there exists a slowly-varying function $\sigma\colon (0,\infty) \to (0,\infty)$ and a map $k \mapsto \alpha_k$ satisfying the following properties:*

(i) *for all k and x, $\mathrm{cost}_k(x) = x^{\alpha_k}\sigma(x)$;*

(ii) *for all k, $\alpha_k \in [0,k]$;*

(iii) *for all k and x, $\mathrm{cost}_k(x) \leq x^k$;*

(iv) *either $\liminf_{k\to\infty} \alpha_k > 0$ or there exists x_0 such that for all k, cost_k is non-decreasing on (x_0,∞);*

(v) *there is an algorithm to compute $\lfloor \alpha_k \rfloor$ in time $\mathcal{O}(k^{9k})$.*

We say that k is the parameter *of* cost*, α is the* index *of* cost*, and σ is the* slowly-varying component *of* cost*.*

Point (i) is the main restriction, and the one we have been discussing until now. Points (ii) and (iii) have already been discussed in the introduction. In the colourful case we will need to compute $\lfloor \alpha_k \rfloor$, so point (v) avoids an additive term in the running time; the precise choice of k^{9k} is purely for technical convenience. Finally, the purpose of point (iv) is to guarantee monotonicity (together with point (i) and Lemma 8(iv)). Requiring point (iv) is unlikely to affect applications of our results – typically such applications would satisfy either $\alpha_k = 0$ (for tracking total query count) or $\alpha_k \geq 1$ (for tracking total running time, where query cost is the running time of a decision algorithm which needs to read its entire input).

▶ **Theorem 10.** *Theorems 1 and 2 remain valid when the cost function $\mathrm{cost}_k(x) = x^{\alpha_k}$ is replaced by an arbitrary regularly-varying parameterised cost function. Likewise, Theorems 4 and 6 remain valid when the the running time $\tilde{O}(n^{\alpha_k})$ of the decision algorithm is replaced by an arbitrary regularly-varying parameterised cost function.*

References

1 Raghavendra Addanki, Andrew McGregor, and Cameron Musco. Non-adaptive edge counting and sampling via bipartite independent set queries. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 2:1–2:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.ESA.2022.2`.

2 Simon Apers, Yuvan Efron, PawełGawrychowski, Troy Lee, Sagnif Mukhopadhyay, and Danupon Nanongkai. Cut query algorithms with star contraction. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, 2022, to appear. arXiv:2201.05674.

3 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. *ACM Trans. Algorithms*, 16(4):52:1–52:27, 2020. doi:10.1145/3404867.

4 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Hyperedge estimation using polylogarithmic subset queries. *CoRR*, abs/1908.04196, 2019. arXiv:1908.04196.

5 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Faster counting and sampling algorithms using colorful decision oracle. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 10:1–10:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.STACS.2022.10.

6 Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. Parameterized query complexity of hitting set using stability of sunflowers. In Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao, editors, *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, volume 123 of *LIPIcs*, pages 25:1–25:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ISAAC.2018.25.

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

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

9 Marco Bressan and Marc Roth. Exact and approximate pattern counting in degenerate graphs: New algorithms, hardness results, and complexity dichotomies. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 276–285, 2022. doi:10.1109/FOCS52979.2021.00036.

10 Timothy Chan, Virginia Vassilevska Williams, and Yinzhan Xu. Fredman's trick meets dominance product: Fine-grained complexity of unweighted APSP, 3SUM counting, and more. *CoRR*, abs/2303.14572, 2023. arXiv:2303.14572.

11 Xi Chen, Amit Levi, and Erik Waingarten. Nearly optimal edge estimation with independent set queries. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2916–2935. SIAM, 2020. doi:10.1137/1.9781611975994.177.

12 Holger Dell and John Lapinskas. Fine-grained reductions from approximate counting to decision. *ACM Trans. Comput. Theory*, 13(2):8:1–8:24, 2021. doi:10.1145/3442352.

13 Holger Dell, John Lapinskas, and Kitty Meeks. Approximately counting and sampling small witnesses using a colorful decision oracle. *SIAM J. Comput.*, 51(4):849–899, 2022. doi:10.1137/19m130604x.

14 Holger Dell, John Lapinskas, and Kitty Meeks. Nearly optimal independence oracle algorithms for edge estimation in hypergraphs. *CoRR*, 2024. arXiv:2211.03874, doi:10.48550/arXiv.2211.03874.

15 William Feller. *An introduction to probability theory and its applications. Vol. II.* Second edition. John Wiley & Sons Inc., New York, 1971.

16 Jacob Focke, Leslie Ann Goldberg, Marc Roth, and Stanislav Zivný. Approximately counting answers to conjunctive queries with disequalities and negations. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '22, pages 315–324, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3517804.3526231.

17 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

18 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), March 2007. doi:10.1145/1206035.1206036.

19 Hung Le. Approximate distance oracles for planar graphs with subpolynomial error dependency. In *Proceedings of the 2023 annual ACM-SIAM symposium on discrete algorithms (SODA)*, pages 1877–1904, 2023.

20 Daniel Lokshtanov, Andreas Björklund, Saket Saurabh, and Meirav Zehavi. Approximate counting of k-paths: Simpler, deterministic, and in polynomial space. *ACM Trans. Algorithms*, 17(3):26:1–26:44, 2021. doi:10.1145/3461477.

21 Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Efficient computation of representative weight functions with applications to parameterized counting (extended version). In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 179–198. SIAM, 2021. doi:10.1137/1.9781611976465.13.

22 Kitty Meeks. The challenges of unbounded treewidth in parameterised subgraph counting problems. *Discrete Applied Mathematics*, 198:170–194, 2016. doi:10.1016/j.dam.2015.06.019.

23 Kitty Meeks. Randomised enumeration of small witnesses using a decision oracle. *Algorithmica*, 81(2):519–540, 2019. doi:10.1007/s00453-018-0404-y.

24 Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: Sequential, cut-query, and streaming algorithms. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 496–509, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384334.

25 Moritz Müller. Randomized approximations of parameterized counting problems. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, volume 4169 of *Lecture Notes in Computer Science*, pages 50–59. Springer, 2006. doi:10.1007/11847250_5.

26 Pan Peng and Jiapeng Zhang. Towards a query-optimal and time-efficient algorithm for clustering with a faulty oracle. In *Proceedings of Machine Learning Research (COLT)*, volume 134, pages 1–19, 2021.

27 Cyrus Rashtchian, David P. Woodruff, and Hanlin Zhu. Vector-matrix-vector queries for solving linear algebra, statistics, and graph problems. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 26:1–26:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.26.

28 Jakub Tetek and Mikkel Thorup. Edge sampling and graph parameter estimation via vertex neighborhood accesses. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1116–1129. ACM, 2022. doi:10.1145/3519935.3520059.

29 Marc Thurley. An approximation algorithm for #k-SAT. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, volume 14 of *LIPIcs*, pages 78–87. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.STACS.2012.78.

30 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47:85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

31 R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. doi:10.1137/15M1024524.

32 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. doi:10.1145/3186893.