

Fine-grained reductions from approximate counting to decision

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

JOHN LAPINSKAS, University of Bristol, UK

In this paper, we introduce a general framework for fine-grained reductions of approximate counting problems to their decision versions. (Thus we use an oracle that decides whether any witness exists to multiplicatively approximate the number of witnesses with minimal overhead.) This mirrors a foundational result of Sipser (STOC 1983) and Stockmeyer (SICOMP 1985) in the polynomial-time setting, and a similar result of Müller (IWPEC 2006) in the FPT setting. Using our framework, we obtain such reductions for some of the most important problems in fine-grained complexity: the Orthogonal Vectors problem, 3SUM, and the Negative-Weight Triangle problem (which is closely related to All-Pairs Shortest Path). While all these problems have simple algorithms over which it is conjectured that no polynomial improvement is possible, our reductions would remain interesting even if these conjectures were proved; they have only polylogarithmic overhead, and can therefore be applied to subpolynomial improvements such as the $n^3/\exp(\Theta(\sqrt{\log n}))$ -time algorithm for the Negative-Weight Triangle problem due to Williams (STOC 2014). Our framework is also general enough to apply to versions of the problems for which more efficient algorithms are known. For example, the Orthogonal Vectors problem over $\text{GF}(m)^d$ for constant m can be solved in time $n \cdot \text{poly}(d)$ by a result of Williams and Yu (SODA 2014); our result implies that we can approximately count the number of orthogonal pairs with essentially the same running time.

We also provide a fine-grained reduction from approximate #SAT to SAT. Suppose the Strong Exponential Time Hypothesis (SETH) is false, so that for some $1 < c < 2$ and all k there is an $O(c^k)$ -time algorithm for k -SAT. Then we prove that for all k , there is an $O((c + o(1))^k)$ -time algorithm for approximate # k -SAT. In particular, our result implies that the Exponential Time Hypothesis (ETH) is equivalent to the seemingly-weaker statement that there is no algorithm to approximate #3-SAT to within a factor of $1 + \epsilon$ in time $2^{o(n)}/\epsilon^2$ (taking $\epsilon > 0$ as part of the input).

CCS Concepts: • **Theory of computation** → *Problems, reductions and completeness; Graph algorithms analysis*; • **Mathematics of computing** → *Graph algorithms*.

Additional Key Words and Phrases: Fine-grained complexity, Approximate Counting, Satisfiability

1 INTRODUCTION

It is clearly at least as hard to count objects as it is to decide their existence, and often it is harder. For a concrete example, there is a polynomial-time algorithm to find a perfect matching in a bipartite graph if one exists, but computing the exact number of all perfect matchings is a #P-complete problem [32], which means that solving it in polynomial time would collapse the polynomial-time hierarchy [30]. However, the situation changes substantially if we consider approximate rather than exact counting. For all real ϵ with $0 < \epsilon < 1$, we say that $x \in \mathbb{R}$ is an ϵ -approximation to $N \in \mathbb{R}$ if $|x - N| \leq \epsilon N$ holds. Since the approximation guarantee is multiplicative, computing an ϵ -approximation to N is at least as hard as deciding whether $N > 0$ holds. In fact, these two tasks are often roughly equally hard, and indeed this is true for our example: Jerrum, Sinclair, and Vigoda [20] proved that an ϵ -approximation to the number of perfect matchings in a bipartite graph can be computed in polynomial time. While there is a polynomial-time algorithm to find perfect matchings in bipartite graphs and one to approximately count them, there is still an important

Authors' addresses: Holger Dell, Goethe University Frankfurt, Frankfurt, Germany, IT University of Copenhagen, Copenhagen, Denmark, Basic Algorithms Research Copenhagen (BARC), Copenhagen, Denmark, dell@cs.uni-frankfurt.de; John Lapinskas, University of Bristol, Bristol, UK, john.lapinskas@bristol.ac.uk.

discrepancy: The former algorithm runs in quasi-linear time while the latter runs in time $\varepsilon^{-2} \cdot \tilde{O}(n^{10})$.

This paper is concerned with fine-grained complexity, in which one considers the exact running time of an algorithm rather than broad categories such as polynomial time, FPT time, or subexponential time. Reductions that solve an approximate counting problem by means of an oracle for its decision version have been studied already in various different contexts. Sipser [27] and Stockmeyer [28] proved implicitly that every problem in #P has a polynomial-time randomised ε -approximation algorithm that has access to an NP-oracle; the result is later made explicit by Valiant and Vazirani [33]. In parameterised complexity, Müller [22] proved an analogue of this result for the W-hierarchy: In particular, for every problem in #W[1], there is a randomised algorithm that has access to some W[1]-oracle, runs in time $f(k) \cdot \text{poly}(n, \varepsilon^{-1})$ for some computable $f : \mathbb{N} \rightarrow \mathbb{N}$, and outputs an ε -approximation to the problem. Finally, in the exponential-time setting, Thurley [29] proposed a reduction for k -SAT that implies: If there is an $O^*(2^{(1-\delta)n})$ -time algorithm for k -SAT for some $\delta > 0$, then there is an ε -approximation algorithm for # k -SAT that runs in time $\varepsilon^{-2} \cdot O^*(2^{(1-\delta/2)n})$. (This reduction was later improved by Schmitt and Wanka [25].) Such results are an important foundation of the wider complexity theory of approximate counting initiated by Dyer, Goldberg, Greenhill and Jerrum [12]. However, all of these reductions introduce significant overheads to the running time — they are not fine-grained.

Perhaps the most important polynomial-time problems in fine-grained complexity are orthogonal vectors (OV), 3SUM, and all-pairs shortest paths (APSP). All three problems admit well-studied notions of hardness, in the sense that many problems reduce to them or are equivalent to them under fine-grained reductions, and they are not known to reduce to one another. See Vassilevska Williams [34] for a recent survey. It is not clear what a “canonical” counting version of APSP should be, but it is equivalent to the Negative-Weight Triangle problem (NWT) under subcubic reductions [35], so we consider this instead. We give highly efficient fine-grained reductions from approximate counting to decision for all three problems. All of these results are immediate corollaries of an algorithm which counts edges in a bipartite graph to which it has limited oracle access; this algorithm has several additional applications, including some new approximate counting algorithms for related problems. We discuss our edge-counting framework further in Section 1.1, and describe its applications in Section 1.2 together with a detailed overview of the literature.

The most important exponential-time problem in fine-grained complexity is unequivocally SAT. We provide a fine-grained reduction from approximate # k -SAT to $O(k \log^2 k)$ -SAT as $k \rightarrow \infty$; as a corollary, we show that if the Strong Exponential Time Hypothesis (SETH) is false, then the savings from decision k -SAT as $k \rightarrow \infty$ can be passed on to approximate # k -SAT with subexponential overhead. Our reduction also implies that the Exponential Time Hypothesis (ETH) is equivalent to an approximate counting version. We discuss the reduction and its corollaries further in Section 1.3.

1.1 Approximately Counting Edges in Bipartite Graphs

Let G be a bipartite graph with $G = (U, V, E)$. We consider a computation model where the algorithm is given U and V , and can access the edges of the graph only via its adjacency oracle and its independence oracle:

- The *adjacency oracle* of G is the function $\text{adj}_G : U \times V \rightarrow \{0, 1\}$ such that $\text{adj}_G(u, v) = 1$ if and only if $(u, v) \in E$.
- The *independence oracle* of G is the function $\text{ind}_G : 2^{U \cup V} \rightarrow \{0, 1\}$ such that $\text{ind}_G(S) = 1$ if and only if S is an independent set in G .

Of course, the adjacency oracle can be simulated with the independence oracle by querying sets of two vertices. We distinguish them here, because we wish to think of independence queries as very expensive, and we will use them only polylogarithmically often. Our main result is as follows:

THEOREM 1. *There is a randomised algorithm \mathcal{A} which, given a rational number ε with $0 < \varepsilon < 1$ and oracle access to an n -vertex bipartite graph G , outputs an ε -approximation of $|E(G)|$ with probability at least $2/3$. Moreover, \mathcal{A} runs in time $\varepsilon^{-2} \cdot O(n \log^4 n \log \log n)$ and makes at most $\varepsilon^{-2} \cdot O(\log^5 n \log \log n)$ calls to the independence oracle.*

We prove this result in Section 4. Note that since oracle calls are constant time operations, the adjacency oracle is called at most $\varepsilon^{-2} \cdot \tilde{O}(n)$ times. Moreover, a polynomial factor of ε^{-1} in the running time is to be expected, since the exact value of $|E(G)|$ can be recovered by taking $\varepsilon = 1/(2n^2)$.

In independent work, Beame et al. [3] obtain a result similar to Theorem 1, with an overall running time of $\varepsilon^{-4} \cdot \tilde{O}(1)$ but with no further bound on the number of independence queries used. Thus their result outperforms Theorem 1 when independence queries are fast, and underperforms when they are slow. In all our applications, independence queries are so slow as to dominate our running times; thus substituting Beame et al.'s result for Theorem 1 would yield worse algorithms.

While Theorem 1 is not able to deal with the non-bipartite case at all, Beame et al. [3] present a second algorithm in their paper that is able to approximately count edges in general graphs by using $\tilde{O}(n^{2/3})$ queries to the independence oracle. Recently, Chen, Levi, and Waingarten [10] improve the number of queries to $\tilde{O}(\sqrt{n})$ and prove unconditionally that this is optimal for general graphs.

In work subsequent to this paper, using a different technique, the authors and Meeks [11] were able to remove the adjacency queries from Theorem 1 while retaining a bound of $\varepsilon^{-2} \log^{O(1)} n$ for the number of independence queries. Moreover, they generalise the theorem to k -partite k -uniform hypergraphs, where the bound on the number of queries is at most $\varepsilon^{-2} \log^{O(k)} n$, and extend the result to cover approximately-uniform sampling. This generalisation has consequences for the fine-grained complexity of problems that do not directly correspond to bipartite graphs, such as approximately counting graph motifs. Independently, Bhattacharya et al. [4, 5] generalise Theorem 1 to k -partite k -uniform hypergraphs, obtaining a somewhat weaker bound of $\varepsilon^{-4} \log^{O(k)} n$ on the number of queries. Moreover, Bishnu et al. [6] use the generalised oracle to solve various decision problems in parameterised complexity.

1.2 Corollaries for Problems in P

As described in Section 1, the problems Orthogonal Vectors (OV), 3SUM, and Negative-Weight Triangle (NWT) are central players in the field of fine-grained complexity. All three problems have simple polynomial-time exhaustive-search algorithms over which it is conjectured that no truly polynomial improvement is possible. The same exhaustive search algorithms also solve the canonical counting versions of these problems. Nevertheless it is possible that the decision version has faster algorithms while the exact counting version does not. Our results imply that any improvement to decision algorithms transfers to the approximate counting version of the problem as well, up to polylogarithmic factors in the running time.

In fact, for OV [1] and NWT [37], non-trivial (subpolynomial) improvements over exhaustive search algorithms are already known. Our results transfer these improvements to approximate counting. In the case of the standard version of OV, this turns out to be uninteresting as the derandomisation of [1] due to Chan and Williams [9] already solves the *exact* counting version. However in the case of NWT, we are not aware of improved algorithms for the counting version; using our reduction, we obtain such an algorithm for approximate counting. Our reductions also apply

to several variants of the three central problems, yielding more new algorithms. Notably, for one variant of OV we obtain a quasilinear-time approximate counting algorithm, but all exact counting algorithms require quadratic time under SETH. In the following, we state our results formally.

1.2.1 OV. In the orthogonal vectors problem OV, we are given two lists A and B of zero-one vectors over \mathbb{R}^d , and must determine whether there exists an orthogonal pair $(a, b) \in A \times B$. In #OV, we must instead determine the number of orthogonal pairs. Writing $n = |A| + |B|$, it is easy to see that OV and #OV can both be solved in $O(n^2d)$ operations by iterating over all pairs. The *low-dimension OV conjecture* [14, 36] asserts that in the case where $d = \omega(\log n)$, there is no randomised algorithm that solves OV in time $O(n^{2-\delta})$, for any constant $\delta > 0$. This conjecture is implied by the Strong Exponential Time Hypothesis (SETH) [36], and Abboud, Williams, and Yu [1] proved that it fails when $d = O(\log n)$.

To reduce the approximate version of #OV to OV, we model the instance as a bipartite graph and apply the edge estimation algorithm from Theorem 1. Indeed, the list A becomes the left side of the graph, B the right side, and each orthogonal pair (a, b) becomes an edge. Then approximately counting orthogonal pairs reduces to estimating the number of edges in this graph, adjacency queries take time $O(d)$ and correspond to computing the inner product of two vectors, and independence queries are simulated by invoking the assumed decision algorithm for OV. In this way, in Section 5.2 we obtain the following structural complexity result as a corollary to Theorem 1.

THEOREM 2. *If OV with n vectors in d dimensions has a randomised algorithm that runs in time $T(n, d)$, then there is a randomised ε -approximation algorithm for #OV that runs in time $T(n, d) \cdot \varepsilon^{-2}O(\log^6 n \log \log n)$.*

In particular, if ε^{-1} is at most polylogarithmic in n , Theorem 2 implies we can ε -approximate #OV with only polylogarithmic overhead over decision.

While OV has a non-trivial algorithm [1] with running time $n^{2-1/O(\log(d/\log n))}$ as we mentioned in Section 1.2, it has already been adapted into an exact #OV algorithm with the same running time [9], so Theorem 2 does not yield a new algorithm at the moment. However, any further improvement for the decision version of the problem will immediately translate to a new approximate counting algorithm.

Interestingly, there is a variant of OV for which our method does yield a new algorithm; in this variant, the real zero-one vectors are replaced by arbitrary vectors over a finite field or over the integers modulo m . Even though Williams and Yu [39] did not consider the counting version and their algorithms do not seem to generalise to counting, we can nevertheless use their decision algorithm as a black box to obtain an efficient approximate counting algorithm as a corollary to Theorem 1.

THEOREM 3. *Let $m = p^k$ be a constant prime power. There is a randomised ε -approximation algorithm for #OV over $\text{GF}(m)^d$ with running time $\varepsilon^{-2}d^{(p-1)k} \cdot \tilde{O}(n)$, and for #OV over $(\mathbb{Z}/m\mathbb{Z})^d$ with running time $\varepsilon^{-2}d^{m-1} \cdot \tilde{O}(n)$.*

If ε^{-1} and d are at most polylogarithmic in n , and m is constant, these algorithms run in quasilinear time. Note that under SETH, any exact counting algorithm for #OV over $(\mathbb{Z}/m\mathbb{Z})^d$ requires time $\Omega(n^{2-o(1)})$ [38]; we have therefore proved a separation between approximate and exact counting. (As an aside, this implies that the factor of ε^{-2} in the running time of Theorem 1 cannot be dropped to $\varepsilon^{-1/2+o(1)}$ under SETH.) Williams and Yu showed that their algorithm's dependence on d is close to best possible under SETH, and this hardness result of course applies to approximate counting as well.

1.2.2 3SUM. In the 3SUM problem, we are given three integer lists A , B , and C of total length n and must decide whether there exists a tuple $(a, b, c) \in A \times B \times C$ with $a + b = c$. One popular extension is 3SUM+, due to Vassilevska Williams and Williams [35], which asks for 3SUM to be solved for all inputs (A, B, c) with $c \in C$. However, as we are specifically concerned with counting problems, we instead consider the problem #3SUM, where we must compute the total number of solution tuples (a, b, c) .

It is easy to see that 3SUM and #3SUM can be solved in $\tilde{O}(n^2)$ operations by sorting C and iterating over all pairs in $A \times B$, and it is conjectured [13, 23] that 3SUM admits no $O(n^{2-\delta})$ -time randomised algorithm for any constant $\delta > 0$. This approach is also how we model instances of 3SUM as a bipartite graph in order to do approximate counting. Joining two vertices a and b whenever $a + b \in C$, adjacency queries can be answered efficiently by binary search on the now-sorted list C , and independence queries on a set $S \subseteq A \cup B$ can be answered by the assumed decision algorithm. Analogous to Theorem 2, in Section 5.1 we obtain the following structural result for 3SUM as a corollary to Theorem 1.

THEOREM 4. *If 3SUM with n integers has a randomised algorithm that runs in time $T(n)$, then there is a randomised ε -approximation algorithm for #3SUM that runs in time $T(n) \cdot \varepsilon^{-2} O(\log^6 n \log \log n)$.*

Thus if ε^{-1} is at most polylogarithmic in n , then the approximate counting algorithm in Theorem 4 has only polylogarithmic overhead over decision. Independently of whether or not the 3SUM conjecture is true, we conclude that 3SUM and, say, $\frac{1}{2}$ -approximating #3SUM have the same time complexity up to polylogarithmic factors.

The fastest-known algorithm for 3SUM, due to Baran, Demaine and Pătraşcu [2], has running time $O(n^2 (\log \log n / \log n)^2)$. Theorem 4 does not currently yield improved algorithms for approximating #3SUM as the polylogarithmic speedup factor of $o(\log^3 n)$ over exhaustive search is smaller than the $O(\log^6 n \log \log n)$ cost in our reduction. However, Chan and Lewenstein [8] prove that 3SUM has much faster algorithms when the input is restricted to instances in which elements of one list are somewhat clustered, in a sense made explicit below. (Their algorithm also works for 3SUM+, but not for #3SUM as far as we can tell.) This is an interesting special case with several applications, including monotone multi-dimensional 3SUM with linearly-bounded coordinates — see the introduction of [8] for an overview. Thus by using the algorithm of Chan and Lewenstein as a black box for the independence oracle, we obtain the following algorithm as a corollary to Theorem 1.

THEOREM 5. *For all $\delta > 0$, there is a randomised ε -approximation algorithm with running time $\varepsilon^{-2} \cdot \tilde{O}(n^{2-\delta/7})$ for instances of #3SUM with n integers such that at least one of A , B , or C may be covered by $n^{1-\delta}$ intervals of length n .*

1.2.3 NWT. In the Negative-Weight-Triangle problem, we are given an edge-weighted graph and must decide whether the graph contains a triangle of negative total weight. Vassilevska Williams and Williams [35] prove that NWT is equivalent to APSP under subcubic reductions. An n -vertex instance of NWT and its natural counting version #NWT can be solved in time $O(n^3)$ by exhaustively checking every possible triangle, and it is conjectured [35] that NWT admits no $O(n^{3-\delta})$ -time randomised algorithm for any constant $\delta > 0$.

To reduce approximate #NWT to its decision version NWT, we put all vertices on one side of the bipartite graph and all edges on the other side. Then adjacency queries correspond to testing whether a given vertex and edge together form a triangle of negative weight, and independence queries can be answered by the assumed decision algorithm for NWT. Thus in Section 5.3 we obtain the following structural result for NWT as a corollary to Theorem 1.

THEOREM 6. *If NWT for n -vertex graphs has a randomised algorithm that runs in time $T(n)$, then there is a randomised ε -approximation algorithm for #NWT that runs in time*

$$T(n) \cdot \varepsilon^{-2} O(\log^6 n \log \log n).$$

Thus if ε^{-1} is at most polylogarithmic, our algorithm has only polylogarithmic overhead over decision. It is known [35] that a truly subcubic algorithm for NWT implies that the negative-weight triangles can also be enumerated in subcubic time. While an enumeration algorithm is obviously stronger than an approximate counting algorithm, this reduction has polynomial overhead and so does not imply Theorem 6.

Williams [37] gives an algorithm with subpolynomial improvements over the exhaustive search algorithm. Using this algorithm as a black-box to answer independence queries, we obtain the following algorithm as a corollary to Theorem 1.

THEOREM 7. *There is a randomised ε -approximation algorithm for #NWT which runs in time $\varepsilon^{-2} n^3 / e^{\Omega(\sqrt{\log n})}$ on graphs with n vertices and polynomially bounded edge-weights.*

1.3 Our results for the satisfiability problem

In k -SAT we are given a k -CNF formula with n variables and must decide whether it is satisfiable. In the natural counting version # k -SAT, we must compute the number of satisfying assignments. The phenomenon that decision, approximate counting, and exact counting seem to become progressively more difficult is nicely represented in the literature: The most efficient known 3-SAT algorithms run in time $O(1.308^n)$ for decision (Hertli [15]), in time $O(1.515^n)$ for $\frac{1}{2}$ -approximate counting (Schmitt and Wanka [25]), and in time $O(1.642^n)$ for exact counting (Kutzkov [21]).

Schmitt and Wanka's algorithm is based on an approach of Thurley [29]. They reduce approximate counting to decision in such a way that an $O^*(2^{(1-\delta_k)n})$ -time algorithm for k -SAT is turned into an ε -approximation algorithm for # k -SAT that runs in time $\varepsilon^{-2} \cdot O^*(2^{(1-\delta'_k)n})$ for some $\delta'_k/2 < \delta'_k < \delta_k$. In the most general form of their algorithm, δ'_k depends on a complicated parameterisation and is calculated on an ad hoc basis for $k = 3$ and $k = 4$, so no asymptotics of $\delta'_k - \delta_k$ are available; the slightly weaker form given in Section 4 of their paper satisfies $\delta'_k \rightarrow \delta_k/2$ as $k \rightarrow \infty$. Thus the exponential savings over exhaustive search go down from δ_k for decision to roughly $\delta_k/2$ for approximate counting. For example, in the extreme case that Impagliazzo and Paturi's [17] exponential time hypothesis (ETH) is false and 3-SAT can be solved in time $2^{o(n)}$, their reduction would only yield an exponential-time algorithm for #3-SAT.

Traxler [31] constructs a reduction from approximate counting to decision, in which savings of δ for decision become $\delta - o(1)$ for approximate counting, so by this metric the reduction is efficient. However, this reduction creates clauses of width $\Omega(\log n)$ and so is not suitable for k -SAT when k is a constant.

We adapt the Valiant–Vazirani style approach of Calabro, Impagliazzo, Kabanets, and Paturi [7] to obtain a reduction from approximate # k -SAT to k' -SAT, with a trade-off between keeping k' close to k versus keeping the cost of the reduction low. At the extremes, writing n for the number of variables in the # k -SAT instance, it implies a reduction from approximate # k -SAT to k -SAT with exponential overhead $2^{O(\log^2 k/k)n}$, or a reduction from approximate # k -SAT to $O(k \log^2 k)$ -SAT with subexponential overhead. We formally state this reduction as Theorem 13 in Section 3.

Our reduction yields interesting structural corollaries for ETH and SETH. Recall that SETH is false if and only if there exists some $\delta > 0$ such that k -SAT can be solved in time $O(2^{(1-\delta)n})$ for all constants k . Our reduction implies not only that SETH is equivalent to its approximate counting version (which is also implied by [25] and [29]), but also that the exponential savings δ must be the same:

THEOREM 8. *Let $0 < \delta < 1$. Suppose that for all $k \in \mathbb{N}$, there is a randomised algorithm which runs on n -variable instances of k -SAT in time $O(2^{(1-\delta)n})$. Then for all $\delta' > 0$ and all $k \in \mathbb{N}$, there is a randomised ε -approximation algorithm which runs on n -variable instances of $\#k$ -SAT in time $\varepsilon^{-2} \cdot O(2^{(1-\delta+\delta')n})$.*

By the sparsification lemma [18], ETH is false if and only if k -SAT can be solved in time $O(2^{\delta n})$ for all constant $\delta > 0$ and k . Since approximate counting always implies decision, ETH clearly implies its seemingly-weaker approximate counting formulation. By letting δ increase to 1 in Theorem 8, we see that the converse is also true:

THEOREM 9. *ETH is false if and only if, for every $k \in \mathbb{N}$ and $\delta > 0$, there is a randomised ε -approximation algorithm that runs on n -variable instances of $\#k$ -SAT in time $\varepsilon^{-2} \cdot O(2^{\delta n})$.*

It remains an open and interesting question whether a result analogous to Theorem 8 holds for fixed k , that is, whether deciding k -SAT and approximating $\#k$ -SAT have the same time complexity up to a subexponential factor. Even a small improvement on Theorem 13 would lead to new algorithms for approximate $\#k$ -SAT. Indeed, for large constant k , the best-known decision, $\frac{1}{2}$ -approximate counting, and exact counting algorithms (due to Paturi, Pudlák, Saks, and Zane [24], Schmitt and Wanka [25], and Impagliazzo, Matthews, and Paturi [16], respectively) all have running time $2^{(1-\Theta(1/k))n}$, but with progressively worse constants in the exponent. If our reduction from approximate $\#k$ -SAT to k -SAT could be improved so that the exponential overhead were $2^{o(1/k)}$ instead of $2^{O((\log k)^2/k)}$, this would yield faster approximate counting algorithms for large but constant k .

1.4 Techniques

Our techniques for the CNF-SAT and the fine-grained results are independent from each other.

CNF-SAT results. We first discuss Theorems 8 and 9, which we prove in Section 3. In the polynomial setting, the standard reduction from approximating $\#k$ -SAT to deciding k -SAT is due to Valiant and Vazirani [33], and runs as follows. If a k -CNF formula F has at most $2^{\delta n}$ solutions for some $\delta > 0$, then we use a standard branching algorithm with $O^*(2^{\delta n})$ calls to a k -SAT-oracle to prune the search tree to size $O(2^{\delta n})$. Otherwise F has many solutions, and for any $m \in \mathbb{N}$, one may form a new formula F_m by conjoining F with m independently-chosen uniformly random XOR clauses. It is relatively easy to see that as long as the number $\text{SAT}(F)$ of satisfying assignments of F is substantially greater than 2^m , then $\text{SAT}(F_m)$ is concentrated around $2^{-m}\text{SAT}(F)$. By choosing m appropriately, one may reduce $\text{SAT}(F_m)$ to below $2^{\delta n}$ and thus compute $\text{SAT}(F_m)$ exactly, then multiply it by 2^m to obtain an estimate for $\text{SAT}(F)$.

Unfortunately, this argument requires modification in the exponential setting. If F has n variables, then each uniformly random XOR has length $\Theta(n)$ and therefore cannot be expressed as a k -CNF formula without introducing $\Omega(n)$ new variables. It follows that (for example) $F_{\lfloor n/2 \rfloor}$ will contain $\Theta(n^2)$ variables. This blowup is acceptable in a polynomial setting, but not an exponential one — for example, given a $\Theta(2^{n^{2/3}})$ -time algorithm for k -SAT, it would yield a useless $\Theta(2^{n^{4/3}})$ -time randomised approximate counting algorithm for $\#k$ -SAT. We can afford to add only constant-length XORs, which do not in general result in concentration in the number of solutions.

We therefore make use of a hashing scheme developed by Calabro, Impagliazzo, Kabanets, and Paturi [7] for a related problem, that of reducing k -SAT to Unique- k -SAT. They choose a $2s$ -sized subset of $[n]$ uniformly at random, where s is a large constant, then choose variables independently at random within that set. This still does not yield concentration in the number of solutions of F_m , but it turns out that the variance is sufficiently low that we can remedy this by summing over many slightly stronger independently-chosen hashes.

Fine-grained results. We now sketch the proof of Theorem 1, which we prove in Section 4. Given a bipartite graph with $G = (U, V, E)$ and $X \subseteq V$, we write $\partial(X)$ for the number of edges incident to X . For all $X \subseteq V$, we may halve $\partial(X)$ in expectation simply by removing half the vertices in X chosen independently at random. Moreover, if $\partial(X)$ is sufficiently small, we may use binary search to efficiently determine $\partial(X)$ exactly. Thus, as with Theorems 8 and 9, we might hope to implement the classical approach of Valiant and Vazirani [33]; start with $X = V$ (so that $\partial(X) = e(G)$), repeatedly approximately halve $\partial(X)$ until it is small enough to determine exactly, then multiply by the appropriate power of 2 and output the result.

Unfortunately, this naive algorithm may fail. For example, if the non-isolated vertices of G form a star whose central vertex lies in V , then the new value of $\partial(X)$ is clearly not concentrated around its expectation; it is either unchanged or reduced to zero. In Lemma 19, we show using martingale techniques that this is essentially the only way things can go wrong. We say X is *balanced* if no single vertex in X is incident to a large proportion of the edges in $G[U \cup X]$ (see Definition 18), and Lemma 19 shows that if X is balanced then with high probability we can approximately halve $\partial(X)$ by deleting half of X uniformly at random.

We therefore proceed by finding a small set of vertices which “unbalances” X if one exists, approximately counting the edges incident to them, and removing them from X . We repeat this process as necessary until X becomes balanced, then delete half of what remains. At the end, we approximate $e(G)$ by taking an appropriate linear combination of our edge counts at each stage. However, since our access to the graph is limited, it is non-trivial to find the “unbalancing” vertices. We must also show that we do not remove too many vertices in this way, as finding edges by brute force is computationally expensive. Our algorithm is essentially given by EdgeCount on p. 16, with some trivial modifications as described in the proof of Theorem 1.

2 PRELIMINARIES

2.1 Notation

We write \mathbb{N} for the set of all positive integers. For a positive integer n , we use $[n]$ to denote the set $\{1, \dots, n\}$. We use \log or \ln to denote the base- e logarithm, and \lg to denote the base-2 logarithm.

We consider graphs G to be undirected, and write $e(G) = |E(G)|$. For all $v \in V(G)$, we use $N(v)$ to denote the neighbourhood $\{w \in V(G) : \{v, w\} \in E(G)\}$ of v . For all $X \subseteq V(G)$, we define $N(X) = \bigcup_{v \in X} N(v)$. We define $\partial(X)$ to be the size of the edge boundary of X , that is, $\partial(X) = |\{e \in E(G) \mid |e \cap X| = 1\}|$. For convenience, we shall generally present bipartite graphs G as a triple (U, V, E) in which (U, V) is a partition of $V(G)$ and $E \subseteq U \times V$.

When stating quantitative bounds on running times of algorithms, we assume the standard word-RAM machine model with logarithmic-sized words. We assume that lists and functions in the problem input are presented in the natural way, that is, as an array using at least one word per entry, and we assume that numerical values such as the edge weights in NWT are given in binary. We shall write $f(x) = \tilde{O}(g(x))$ when for some constant $c \in \mathbb{R}$, $f(x) = O((\log x)^c g(x))$ as $x \rightarrow \infty$. Similarly, we write $f(x) = O^*(g(x))$ when for some constant $c \in \mathbb{R}$, $f(x) = O(x^c g(x))$ as $x \rightarrow \infty$.

We require our problem inputs to be given as finite binary strings, and write Σ^* for the set of all such strings. A *randomised approximation scheme* for a function $f : \Sigma^* \rightarrow \mathbb{N}$ is a randomised algorithm that takes as input an instance $x \in \Sigma^*$ and a rational error tolerance $0 < \varepsilon < 1$, and outputs a rational number z (a random variable depending on the “coin tosses” made by the algorithm) such that, for every instance x , $\mathbb{P}((1 - \varepsilon)f(x) \leq z \leq (1 + \varepsilon)f(x)) \geq 2/3$. All of our approximate counting algorithms will be randomised approximation schemes.

2.2 Probability theory

We use some results from probability theory, which we collate here for reference. First, we state Chebyshev's inequality.

LEMMA 10. *Let X be a real-valued random variable with mean μ and let $t > 0$. Then*

$$\mathbb{P}(|X - \mu| \geq t) \leq \frac{\text{Var}(X)}{t^2}. \quad \square$$

We also use the following concentration result due to McDiarmid [26].

LEMMA 11. *Let f be a real function of independent random variables X_1, \dots, X_m , and let $\mu = \mathbb{E}(f(X_1, \dots, X_m))$. Let $c_1, \dots, c_m \geq 0$ such that, for all $i \in [m]$ and all pairs $(\mathbf{x}, \mathbf{x}')$ differing only in the i th coordinate, we have $|f(\mathbf{x}) - f(\mathbf{x}')| \leq c_i$. Then for all $t > 0$,*

$$\mathbb{P}(|f(X_1, \dots, X_m) - \mu| \geq t) \leq 2e^{-2t^2 / \sum_{i=1}^m c_i^2}. \quad \square$$

Finally, we use the following Chernoff bounds, proved in (for example) Corollaries 2.3–2.4 and Remark 2.11 of Janson, Łuczak and Ruciński [19].

LEMMA 12. *Let X be a binomial or hypergeometric random variable with mean μ .*

(i) *For all ε with $0 < \varepsilon \leq \frac{3}{2}$, we have $\mathbb{P}(|X - \mu| \geq \varepsilon\mu) \leq 2e^{-\varepsilon^2\mu/3}$.*

(ii) *For all t with $t \geq 7\mu$, we have $\mathbb{P}(X \geq t) \leq e^{-t}$.* □

3 FROM DECISION TO APPROXIMATE COUNTING CNF-SAT

In this section we prove our results for the satisfiability of CNF formulae, formally defined as follows.

Problem k -SAT expects as input: A k -CNF formula F .

Task: Decide if F is satisfiable.

Problem $\#k$ -SAT expects as input: A k -CNF formula F .

Task: Compute the number $\text{SAT}(F)$ of satisfying assignments of F .

We also define a technical intermediate problem. For all $s \in \mathbb{N}$, we say that a matrix A is s -sparse if every row of A contains at most s non-zero entries. In the following definition, $k \in \mathbb{N}$ and $s \in \mathbb{N}$ are constants.

Problem $\Pi_{k,s}$ expects as input: An n -variable Boolean formula F of the form $F(\mathbf{x}) = F'(\mathbf{x}) \wedge (A\mathbf{x} = \mathbf{b})$. Here F' is a k -CNF formula, A is an s -sparse $m \times n$ matrix over $\text{GF}(2)$ with $0 \leq m \leq n$, and $\mathbf{b} \in \text{GF}(2)^m$.

Task: Decide if F is satisfiable.

We define the *growth rate* $\pi_{k,s}$ of $\Pi_{k,s}$ as the infimum over all $\beta > 0$ such that $\Pi_{k,s}$ has a randomised algorithm that runs in time $O^*(2^{\beta n})$ and outputs the correct answer with probability at least $2/3$. Our main reduction is encapsulated in the following theorem.

THEOREM 13. *Let $k \in \mathbb{N}$ with $k \geq 2$, let $0 < \delta < 1$, and let $s \geq 120 \lg^2(6/\delta)/\delta$. Then there is a randomised approximation scheme for $\#k$ -SAT which, when given an n -variable formula F and approximation error parameter ε , runs in time $\varepsilon^{-2} \cdot O(2^{(\pi_{k,s} + \delta)n})$.*

Before we prove this theorem, let us derive Theorems 8 and 9 as immediate corollaries. In both cases, we use the fact that the condition $Ax = b$ can be expressed as an s -CNF formula with $m2^{s-1}$ clauses, and thus $\pi_{k,s} \leq \pi_{\max\{k,s\},0}$ holds for all constant k, s .

THEOREM 8 (RESTATED). *Let $0 < \delta < 1$. Suppose that for all $k \in \mathbb{N}$, there is a randomised algorithm which runs on n -variable instances of k -SAT in time $O(2^{(1-\delta)n})$. Then for all $\delta' > 0$ and all $k \in \mathbb{N}$, there is a randomised ε -approximation algorithm which runs on n -variable instances of $\#k$ -SAT in time $\varepsilon^{-2} \cdot O(2^{(1-\delta+\delta')n})$.*

PROOF. Let $\delta > 0$ be as specified in the theorem statement. Then for all constant $k, s \in \mathbb{N}$, we have $\pi_{k,s} \leq \pi_{\max\{k,s\},0} \leq 1 - \delta$. The result follows by Theorem 13 with $s = 120 \lg^2(6/\delta')/\delta'$. \square

THEOREM 9 (RESTATED). *ETH is false if and only if, for every $k \in \mathbb{N}$ and $\delta > 0$, there is a randomised ε -approximation algorithm that runs on n -variable instances of $\#k$ -SAT in time $\varepsilon^{-2} \cdot O(2^{\delta n})$.*

PROOF. The backward implication is immediate: Any randomised $\frac{1}{2}$ -approximation scheme for $\#3$ -SAT is able to decide 3-SAT with success probability at least $2/3$. For the forward implication, assume ETH is false. By the sparsification lemma [18, Lemma 10], we then have $\pi_{k,0} = 0$ for all $k \in \mathbb{N}$. Hence for all $k, s \in \mathbb{N}$, we obtain $\pi_{k,s} \leq \pi_{\max\{k,s\},0} = 0$. The result now follows by Theorem 13. \square

3.1 Proof of Theorem 13

Given access to an oracle that decides satisfiability queries, we can compute the exact number of solutions of a formula with few solutions using a standard self-reducibility argument given below (see also [29, Lemma 3.2]).

Algorithm $\text{CountFew}(F, a)$: *Given an instance F of $\Pi_{k,s}$ on n variables, $a \in \mathbb{N}$, and access to an oracle for $\Pi_{k,s}$, this algorithm computes $\text{SAT}(F)$ if $\text{SAT}(F) \leq a$; otherwise it outputs FAIL.*

- 1 (Query the oracle) If F is unsatisfiable, return 0.
- 2 (No variables left) If F contains no variables, return 1.
- 3 (Branch and recurse) Let F_0 and F_1 be the formulae obtained from F by setting the first free variable in F to 0 and 1, respectively. If $\text{CountFew}(F_0, a) + \text{CountFew}(F_1, a)$ is at most a , then return this sum; otherwise abort the entire computation and return FAIL.

LEMMA 14. *CountFew is correct and runs in time at most $(\min\{a, \text{SAT}(F)\} + 1) \cdot \tilde{O}(|F|)$. Moreover, each oracle query is a formula with at most n variables.*

PROOF. The correctness of CountFew follows by induction from $\text{SAT}(F) = \text{SAT}(F_0) + \text{SAT}(F_1)$. For the running time, consider the recursion tree of CountFew on inputs F and a . At each vertex, the algorithm takes time at most $\tilde{O}(|F|)$ to compute F_0 and F_1 , and it issues a single oracle call. For convenience, we call the leaves of the tree at which CountFew returns 0 in Step 1 or 1 in Step 2 the *0-leaves* and *1-leaves*, respectively. Let x be the number of 1-leaves. Each non-leaf is on the path from some 1-leaf to the root, otherwise it would be a 0-leaf. There are at most x such paths, so there are at most nx non-leaf vertices in total. Finally, every 0-leaf has a sibling which is not a 0-leaf, or its parent would be a 0-leaf, so there are at most $(n+1)x$ 0-leaves in total. Overall, the tree has at most $4nx$ vertices. An easy induction using Step 3 implies that $x \leq 2a$, and certainly $x \leq \text{SAT}(F)$, so the claimed running time is correct. \square

When our input formula F has too many solutions to apply CountFew efficiently, we first reduce the number of solutions by hashing. In particular, we use the same hash functions as Calabro et al. [7]; they are based on random sparse matrices over $\text{GF}(2)$ and formally defined as follows:

DEFINITION 15. Let $s, m, n \in \mathbb{N}$. An (s, m, n) -hash is a random $m \times n$ matrix A over $\text{GF}(2)$ defined as follows. For each row $i \in [m]$, let R_i be a uniformly random size- s subset of $[n]$. Then for all $i \in [m]$ and all $j \in R_i$, we choose values $A_{i,j} \in \text{GF}(2)$ independently and uniformly at random, and set all other entries of A to zero.

For intuition, suppose that F is an n -variable k -CNF formula, S is the set of satisfying assignments of F , and $|S| > 2^{\delta n}$ holds for some small $\delta > 0$. It is easy to see that, for all $m, s \in \mathbb{N}$ and uniformly random $\mathbf{b} \in \text{GF}(2)^m$, if A is an (s, m, n) -hash, then the number X of satisfying assignments of $F(\mathbf{x}) \wedge (A\mathbf{x} = \mathbf{b})$ has expected value $|S|/2^m$. (See Lemma 16.) If X were concentrated around its expectation, then by choosing an appropriate value of m , we could reduce the number of solutions to at most $2^{\delta n}$, apply CountFew to count them exactly, then multiply the result by 2^m to obtain an approximation to $|S|$. This is the usual approach pioneered by Valiant and Vazirani [33].

In the exponential setting, however, we can only afford to take $s = O(1)$, which means that X is not in general concentrated around its expectation. In [7], only very limited concentration was needed, but we require strong concentration. To achieve this, rather than counting satisfying assignments of a single formula $F(\mathbf{x}) \wedge (A\mathbf{x} = \mathbf{b})$, we will sum over many such formulae. We first bound the variance of an individual (s, m, n) -hash when s and S are suitably large. Our analysis here is similar to that of Calabro et al. [7], although they are concerned with lower-bounding the probability that at least one solution remains after hashing and do not give bounds on variance.

LEMMA 16. Let $\delta \in \mathbb{R}$ with $0 < \delta < \frac{1}{6}$ and let $s, m, n \in \mathbb{N}$. Suppose $m \leq n$ and $s \geq 20 \lg^2(1/\delta)/\delta$. Let $S \subseteq \text{GF}(2)^n$ and suppose $|S| \geq 2^{m+\delta n}$. Let A be an (s, m, n) -hash, and let $\mathbf{b} \in \text{GF}(2)^m$ be uniformly random and independent of A . Let $S' = \{\mathbf{x} \in S : A\mathbf{x} = \mathbf{b}\}$. Then $\mathbb{E}(|S'|) = 2^{-m}|S|$ and $\text{Var}(|S'|) \leq |S|^2 2^{\delta n/8-2m}$.

PROOF. For each $\mathbf{x} \in \text{GF}(2)$, let $I_{\mathbf{x}}$ be the indicator variable of the event $A\mathbf{x} = \mathbf{b}$. Exposing A implies $\mathbb{P}(I_{\mathbf{x}}) = 2^{-m}$ for all $\mathbf{x} \in \text{GF}(2)^n$, and hence

$$\mathbb{E}(|S'|) = \sum_{\mathbf{x} \in S} \mathbb{P}(I_{\mathbf{x}}) = 2^{-m}|S|.$$

We now bound the second moment. We have

$$\begin{aligned} \mathbb{E}(|S'|^2) &= \sum_{(\mathbf{x}, \mathbf{y}) \in S^2} \mathbb{E}(I_{\mathbf{x}} I_{\mathbf{y}}) = \sum_{(\mathbf{x}, \mathbf{y}) \in S^2} \mathbb{P}(A\mathbf{x} = A\mathbf{y} = \mathbf{b}) \\ &= \sum_{(\mathbf{x}, \mathbf{y}) \in S^2} \prod_{i=1}^m \mathbb{P}((A\mathbf{x})_i = (A\mathbf{y})_i = \mathbf{b}_i). \end{aligned} \quad (1)$$

When \mathbf{x} and \mathbf{y} are fixed, the events in (1) are identically distributed and we write $p_{\mathbf{x}, \mathbf{y}} = \mathbb{P}(\mathbf{a}^T \mathbf{x} = \mathbf{a}^T \mathbf{y} = b)$, where $b \in \{0, 1\}$ is sampled uniformly at random and $\mathbf{a} \in \{0, 1\}^n$ is sampled by first sampling a size- s set $R \subseteq \{1, \dots, n\}$ and then setting the bits \mathbf{a}_j uniformly for $j \in R$, and $\mathbf{a}_j = 0$ for $j \notin R$. Using this shorthand notation, we split the sum in (1) depending on whether the Hamming distance $d(\mathbf{x}, \mathbf{y})$ between the vectors is at most αn or larger, for some parameter $\alpha < \frac{1}{2}$ specified later.

$$\mathbb{E}(|S'|^2) = \sum_{(\mathbf{x}, \mathbf{y}) \in S^2} p_{\mathbf{x}, \mathbf{y}}^m = \sum_{\substack{(\mathbf{x}, \mathbf{y}) \in S^2 \\ d(\mathbf{x}, \mathbf{y}) \leq \alpha n}} p_{\mathbf{x}, \mathbf{y}}^m + \sum_{\substack{(\mathbf{x}, \mathbf{y}) \in S^2 \\ d(\mathbf{x}, \mathbf{y}) > \alpha n}} p_{\mathbf{x}, \mathbf{y}}^m. \quad (2)$$

We now provide upper bounds for these two sums. For the first sum, let us write $h : [0, 1] \rightarrow [0, 1]$ for the binary entropy function $h(\alpha) = -\alpha \lg \alpha - (1 - \alpha) \lg(1 - \alpha)$; it is known that the Hamming ball of radius αn around a binary vector \mathbf{x} contains at most $2^{h(\alpha)n}$ binary vectors \mathbf{y} . Thus the first

sum is bounded by $|S|2^{h(\alpha)n} \max\{p_{\mathbf{x},\mathbf{y}}^m\}$. To bound the maximum, note by exposing \mathbf{a} that $p_{\mathbf{x},\mathbf{y}} \leq \frac{1}{2}$ holds for all \mathbf{x}, \mathbf{y} . Thus, the first sum in (2) is bounded by $|S|2^{h(\alpha)n-m}$.

The second sum in (2) is at most $|S|^2 \max\{p_{\mathbf{x},\mathbf{y}}^m : d(\mathbf{x}, \mathbf{y}) > \alpha n\}$, and so it remains to bound $p_{\mathbf{x},\mathbf{y}}$ for vectors \mathbf{x} and \mathbf{y} whose distance is more than αn . Write $\mathbf{x}_R \in \text{GF}(2)^R$ for the projection of \mathbf{x} to the coordinates of R . Conditioning on the event $\mathbf{x}_R = \mathbf{y}_R$, we get

$$\begin{aligned} p_{\mathbf{x},\mathbf{y}} &= \mathbb{P}\left(\mathbf{a}^T \mathbf{x} = \mathbf{a}^T \mathbf{y} = b \mid \mathbf{x}_R \neq \mathbf{y}_R\right) \cdot \mathbb{P}(\mathbf{x}_R \neq \mathbf{y}_R) \\ &\quad + \mathbb{P}\left(\mathbf{a}^T \mathbf{x} = \mathbf{a}^T \mathbf{y} = b \mid \mathbf{x}_R = \mathbf{y}_R\right) \cdot \mathbb{P}(\mathbf{x}_R = \mathbf{y}_R) \\ &\leq \mathbb{P}\left(\mathbf{a}^T \mathbf{x} = \mathbf{a}^T \mathbf{y} = b \mid \mathbf{x}_R \neq \mathbf{y}_R\right) + \frac{1}{2} \cdot \mathbb{P}(\mathbf{x}_R = \mathbf{y}_R). \end{aligned} \quad (3)$$

We claim that the first summand of (3) is equal to $\frac{1}{4}$ and the second is bounded above by $\frac{1}{2}e^{-\alpha s}$. Indeed, conditioned on $\mathbf{x}_R \neq \mathbf{y}_R$, there is a coordinate $c \in R$ with $\mathbf{x}_c \neq \mathbf{y}_c$. Without loss of generality, assume $\mathbf{x}_c = 1$ and $\mathbf{y}_c = 0$. Under this conditioning, the events $\mathbf{a}^T \mathbf{x} = \mathbf{a}^T \mathbf{y}$ and $\mathbf{a}^T \mathbf{y} = b$ are actually independent, because \mathbf{a}_c is a uniform bit that only affects the first event and b is a uniform bit that only affects the second. More precisely, after exposing R with $\mathbf{x}_R \neq \mathbf{y}_R$ and \mathbf{a}_j for all $j \in R \setminus \{c\}$, the probability that \mathbf{a}_c and b are set correctly is $\frac{1}{4}$. To bound the second summand of (3), recall that $d(\mathbf{x}, \mathbf{y}) \geq \alpha n$ and $|R| = s$, and observe

$$\mathbb{P}(\mathbf{x}_R = \mathbf{y}_R) \leq \frac{\binom{n-\lceil \alpha n \rceil}{s}}{\binom{n}{s}} \leq (1 - \lceil \alpha n \rceil / n)^s \leq e^{-\alpha s}.$$

Putting the bounds on the terms in (3) together, we arrive at

$$p_{\mathbf{x},\mathbf{y}} \leq \frac{1}{4} + \frac{1}{2}e^{-\alpha s} = \frac{1}{4}(1 + 2e^{-\alpha s}) \leq \frac{1}{4}e^{2e^{-\alpha s}}.$$

This allows us to bound the second moment and thus the variance as well:

$$\text{Var}(|S'|) = \mathbb{E}(|S'|^2) - \mathbb{E}(|S'|)^2 \leq (|S|2^{h(\alpha)n-m} + |S|^2 4^{-m} e^{m \cdot 2e^{-\alpha s}}) - |S|^2 2^{-2m}. \quad (4)$$

By assumption we have $|S| \geq 2^{m+\delta n}$, and thus $|S|^2 2^{-2m} \geq |S|2^{\delta n-m}$. Now we set $\alpha < \frac{1}{2}$ such that $h(\alpha) = \delta$ holds. Since $\delta < \frac{1}{6}$, we have $\alpha = h^{-1}(\delta) \geq \delta / (2 \lg(6/\delta)) \geq \delta / (4 \lg(1/\delta))$. It follows that $\alpha s \geq 5 \lg(1/\delta) \geq 2 \ln(4/\delta)$, and together with (4) we get $\text{Var}(|S'|) \leq |S|^2 e^{\delta^2 m/8} / 2^{2m}$. Since $m \leq n$ and $\delta < 1/\lg(e)$, the result follows. \square

We now state our algorithm for Theorem 13 that reduces from approximate counting for k -SAT to decision for $\Pi_{k,s}$. In the following definition, δ is a rational constant with $0 < \delta < \frac{1}{3}$.

Algorithm ApxToD_δ : Given an n -variable instance F of $\#k$ -SAT, a rational number $\varepsilon \in (0, 1)$, and access to an oracle for $\Pi_{k,s}$ for some $s \geq 40 \lg^2(2/\delta)/\delta$, this algorithm computes a rational number z such that $(1 - \varepsilon)\text{SAT}(F) \leq z \leq (1 + \varepsilon)\text{SAT}(F)$ holds with probability at least $\frac{3}{4}$.

1 (Brute-force on constant-size instances)

If $n/\lg n \leq 8/\delta$, solve the problem by brute force and return the result.

2 (If there are few satisfying assignments, count them exactly)

Let $t = \lceil \delta n/2 + 2 \lg(1/\varepsilon) \rceil$, and apply CountFew to F and $a = 2^{t+\delta n/2}$. Return the result if it is not equal to FAIL.

3 (Try larger and larger equation systems) For each $m \in \{0, \dots, n - t\}$:

a For each $i \in \{1, \dots, 2^t\}$:

- (Prepare query) Independently sample an $(s, m + t, n)$ -hash $A_{m,i}$ and a uniformly random vector $\mathbf{b}_{m,i} \in \text{GF}(2)^{m+t}$. Let $F_{m,i} = F(\mathbf{x}) \wedge (A_{m,i}\mathbf{x} = \mathbf{b}_{m,i})$.

- (Ask oracle using subroutine) Let $z_{m,i}$ be the output of $\text{CountFew}(F_{m,i}, 4a)$.
 - (Bad randomness or m too small) If $z_{m,i} = \text{FAIL}$ or if $\sum_{j=1}^i z_{m,j} > 4a$, then go to the next m in the outer for-loop.
- b (Return our estimate) Return $z = 2^m \overline{\sum_{i=1}^{2^t} z_{m,i}}$.

LEMMA 17. ApXToD_δ is correct for all $\delta \in (0, \frac{1}{3})$ and runs in time at most $\varepsilon^{-2} \cdot O^*(2^{\delta n})$. Moreover, the oracle is only called on instances with at most n variables.

PROOF. Let F be a k -CNF formula on n variables and let $\varepsilon \in (0, 1)$. For the running time, note that Step 1 takes time $O(2^{1/\delta}) = O(1)$, Step 2 takes time at most $O^*(a)$ by Lemma 14. By the same lemma, each invocation of CountFew on input $F_{m,i}$ in 3 takes time $O^*(\min\{z_{m,i}, a\} + 1)$. Moreover, the outer loop in Step 1 is run at most $n - t$ times, and for each fixed m , executing Step 3a in its entirety takes time at most $O^*(a)$ due to the check whether $\sum_{j=1}^i z_{m,k} > 4a$ holds. Thus the overall running time of the algorithm is $O^*(a) \leq O^*(\varepsilon^{-2} 2^{\delta n})$ as required.

It remains to prove the correctness of the algorithm. If it terminates at Step 1 or Step 2, then correctness is immediate from Lemma 14. Suppose not, so that $n/\lg n > 8/\delta$ holds, and the set S of solutions of F satisfies $|S| \geq 2^{t+\delta n/2}$. Let $M = \max\{m \in \mathbb{Z} : |S| \geq 2^{m+t+\delta n/2}\}$, and note that $0 \leq M \leq n - t$ and $|S| \leq 2^{M+t+\delta n/2+1}$. The formulas $F_{m,i}$ are oblivious to the execution of the algorithm, so for the analysis we may view them as being sampled in advance. Let $S_{m,i}$ be the set of solutions to $F_{m,i}$. For each m with $0 \leq m \leq M$, let \mathcal{E}_m be the following event:

$$\left| \sum_{i=1}^{2^t} |S_{m,i}| - 2^{-m} |S| \right| \leq 2^{-m-(t-\delta n/2)/2} \cdot |S|.$$

Thus \mathcal{E}_m implies $\left| 2^m \sum_{i=1}^{2^t} |S_{m,i}| - |S| \right| \leq \varepsilon |S|$. By Lemma 16 applied with $\delta/2$ in place of δ and $m + t$ in place of m , for all $0 \leq m \leq M$ and $1 \leq i \leq 2^t$, we have $\mathbb{E}(|S_{m,i}|) = 2^{-m-t} |S|$ and $\text{Var}(|S_{m,i}|) \leq |S|^2 2^{\delta n/16-2m-2t}$. Since the $S_{m,i}$'s are independent, it follows by Lemma 10 that

$$\mathbb{P}(\mathcal{E}_m) \geq 1 - \frac{2^t \cdot |S|^2 2^{\delta n/16-2m-2t}}{2^{-2m-t+\delta n/2} |S|^2} \geq 1 - 2^{-\delta n/4} \geq 1 - 1/n^2.$$

Thus a union bound implies that, with probability at least $3/4$, the event \mathcal{E}_m occurs for all m with $0 \leq m \leq M$ simultaneously. Suppose now that this happens. Then in particular, we have

$$\sum_{i=1}^{2^t} |S_{M,i}| \leq (1 + \varepsilon) 2^{-M} |S| \leq 2^{t+\delta n/2+2}.$$

But then, if ApXToD_δ reaches iteration $m = M$, none of the calls to CountFew fail in this iteration and we have $z_{M,i} = |S_{M,i}|$ for all $i \in \{1, \dots, 2^t\}$. Thus ApXToD_δ returns some estimate z while $m \leq M$. Moreover, since \mathcal{E}_m occurs, this estimate satisfies $(1 - \varepsilon)|S| \leq z \leq (1 + \varepsilon)|S|$ as required. Thus ApXToD_δ behaves correctly with probability at least $3/4$, and the result follows. \square

THEOREM 13 (RESTATED). Let $k \in \mathbb{N}$ with $k \geq 2$, let $0 < \delta < 1$, and let $s \geq 120 \lg^2(6/\delta)/\delta$. Then there is a randomised approximation scheme for $\#k$ -SAT which, when given an n -variable formula F and approximation error parameter ε , runs in time $\varepsilon^{-2} \cdot O(2^{(\pi_{k,s} + \delta)n})$.

PROOF. If $\varepsilon < 2^{-n}$, then we solve the $\#k$ -SAT instance exactly by brute force in time $O^*(\varepsilon^{-1})$, so suppose $\varepsilon \geq 2^{-n}$. By the definition of $\pi_{k,s}$, there exists a randomised algorithm for $\Pi_{k,s}$ with failure probability at most $1/3$ and running time at most $O^*(2^{(\pi_{k,s} + \delta/3)n})$. By Lemma 12(i), for any constant C , by applying this algorithm $\lg(1/\varepsilon) \cdot O(n) = O(n^2)$ times and outputting the majority answer, we may reduce the failure probability to at most $\varepsilon^2/Cn2^{\delta n/3}$. We apply $\text{ApXToD}_{\delta/3}$ to F

and ε , using the randomized algorithm for $\Pi_{k,s}$ in place of the $\Pi_{k,s}$ -oracle. If we take C sufficiently large, then by Lemma 17 and a union bound, the overall failure probability is at most $1/3$, and the running time is $\varepsilon^{-2} \cdot O^*(2^{(\pi_{k,s}+2\delta/3)n}) = \varepsilon^{-2} \cdot O(2^{(\pi_{k,s}+\delta)n})$ as required. \square

4 APPROXIMATELY COUNTING EDGES IN BIPARTITE GRAPHS

In this section, we prove our main result, Theorem 1. Recall from Section 1.1 that it consists of an algorithm that is given access to a bipartite graph via an adjacency oracle and an independence oracle. Throughout this section, we fix $G = (U, V, E)$ and $\varepsilon > 0$ as the input to our edge-counting algorithm, and we define $n = |U \cup V|$.

4.1 Random subsets of balanced sets

A set $X \subseteq V$ is *balanced* if the graph $G[U, X]$ is not “star-like”, with a large proportion of edges incident to a single vertex in X . We formally define this notion, and show that if X' is a uniformly random subset of a balanced set X , then $\partial(X) \approx 2\partial(X')$ holds with suitably high probability.

DEFINITION 18. *For any real ξ with $0 < \xi \leq 1$, a set $X \subseteq V$ is ξ -balanced if every vertex in X has degree at most $\xi\partial(X)$.*

LEMMA 19. *Let $X \subseteq V$ be a set and let $X' \subseteq X$ be a random subset formed by including each vertex of X independently with probability $\frac{1}{2}$.*

- (i) *With probability at least $1 - 2 \exp(-|X|/24)$, we have $|X'| \leq \frac{3}{4}|X|$.*
- (ii) *Let γ, ξ be reals with $0 < \xi \leq 1$ and $0 < \gamma \leq \frac{1}{2}$. If X is ξ -balanced, then with probability at least $1 - 2 \exp(-2\gamma^2/\xi)$, we have*

$$\left(\frac{1}{2} - \gamma\right) \cdot \partial(X) \leq \partial(X') \leq \left(\frac{1}{2} + \gamma\right) \cdot \partial(X).$$

PROOF. For the first claim, note that $\mathbb{E}(|X'|) = |X|/2$ holds, and thus by Lemma 12(i) we have

$$\mathbb{P}(|X'| \geq \frac{3}{4} \cdot |X|) \leq \mathbb{P}(|X'| - \frac{1}{2} \cdot |X| \geq \frac{1}{4} \cdot |X|) \leq 2e^{-|X|/24}.$$

Now we prove the second claim. For each vertex $v \in X$, let I_v be the indicator random variable of the event $v \in X'$. Then $\partial(X')$ is a function of $\{I_v : v \in X\}$, and changing a single indicator variable I_v alters $\partial(X')$ by exactly $d(v)$. Moreover, $\mathbb{E}(\partial(X')) = \partial(X)/2$. It therefore follows by Lemma 11 that

$$\mathbb{P}\left(|\partial(X') - \frac{1}{2} \cdot \partial(X)| \geq \gamma \cdot \partial(X)\right) \leq 2 \exp\left(\frac{-2\gamma^2\partial(X)^2}{\sum_{v \in X} d(v)^2}\right). \quad (5)$$

Since X is ξ -balanced, we have $\sum_{v \in X} d(v)^2 \leq \xi\partial(X) \cdot \sum_{v \in X} d(v) = \xi\partial(X)^2$. With (5), the claimed upper bound of $2 \exp(-2\gamma^2/\xi)$ on the error probability follows. \square

In using Lemma 19, we will take $\gamma = \Theta(\varepsilon/\log n)$ and $\xi = \Theta(\gamma^2/\log \log n)$. To motivate this choice, consider the following toy argument:

Suppose simplistically that Lemma 19(ii) was true for all sets, not just for balanced sets, and that ξ could be chosen arbitrarily. We will see later (using the `SampleNeighbours` algorithm defined in Section 4.2) that, if $\partial(X)$ is small, we can quickly determine it exactly. In this situation, the following algorithm would estimate $e(G)$: start with $X_0 = V$. Given X_i , check whether $\partial(X_i)$ is small enough to determine exactly. If so, output $2^i\partial(X_i)$. If not, form X_{i+1} from X_i by including each element independently with probability $\frac{1}{2}$. Let X_t be the final set formed this way. By Lemma 19(i), we have $t = O(\log n)$ with high probability. By our supposed simplistic version of Lemma 19(ii), we have $\partial(X_t) \in (1 \pm \gamma)^t \partial(X_0)/2^t = (1 \pm \gamma)^t e(G)/2^t$; thus the algorithm gives a valid ε -approximation whenever $(1 \pm \gamma)^t \subseteq (1 \pm \varepsilon)$. We have $(1 \pm \gamma)^t \subseteq 1 \pm 4t\gamma$ for sufficiently small γ , so this holds for $\gamma = O(\varepsilon/\log n) = O(\varepsilon/t)$. Finally, using a union bound together with the fact that $t = O(\log n)$

holds with high probability, Lemma 19(ii) holds at each stage with probability at least $1 - O(\log n) \cdot \exp(-2\gamma^2/\xi)$; this can be made arbitrarily large by taking $\xi = O(\gamma^2/\log \log n)$.

Of course, Lemma 19(ii) is not true for all sets — it fails badly if $G[U, X]$ is a star, for example. While the above argument does not use independence queries at all, we will need them to deal with unbalanced sets.

4.2 Estimating vertex degrees

In order to test whether a set X is balanced and thus whether taking a uniformly random subset of X will give a good approximation of $\partial(X)$ via Lemma 19, we will efficiently approximate the *relative degrees* $d(v)/|N(X)|$ for all $v \in X$. To this end, we will use independence queries to uniformly sample a random subset $Y \subseteq N(X)$ of a given size y . We show that, with high probability, the random variable $|N(v) \cap Y|/|Y|$ is a $\frac{1}{2}$ -approximation of the relative degree unless the relative degree is smaller than $\xi/140$, in which case $|N(v) \cap Y|/|Y|$ is no larger than $\xi/20$.

LEMMA 20. *Let $X \subseteq V$ and let $y \in \mathbb{N}$ with $y \leq |N(X)|$. Let $Y \subseteq N(X)$ be a uniformly-random size- y subset of $N(X)$. Let $v \in X$ be a vertex and write*

$$\delta(v) = \frac{|N(v)|}{|N(X)|} \quad \text{and} \quad \tilde{\delta}(v) = \frac{|N(v) \cap Y|}{|Y|}.$$

Let $\xi > 0$. If $\delta(v) \geq \xi/140$, then with probability at least $1 - 2 \exp(-\xi y/2000)$, the number $\tilde{\delta}(v)$ is a $\frac{1}{2}$ -approximation of $\delta(v)$. On the other hand, if $\delta(v) \leq \xi/140$, then with probability at least $1 - 2 \exp(-\xi y/20)$, we have $\tilde{\delta}(v) \leq \xi/20$.

PROOF. The random variable $|N(v) \cap Y|$ follows a hypergeometric distribution with mean $\mu_v = \delta(v) \cdot y$. By Lemma 12(i), we have

$$\mathbb{P}\left(\left||N(v) \cap Y| - \mu_v\right| \geq \frac{\mu_v}{2}\right) \leq 2 \exp(-\mu_v/12).$$

If $\delta(v) \geq \xi/140$ and thus $\mu_v \geq \xi y/140$, this immediately implies the first claim. Similarly, if $\delta(v) \leq \xi/140$ and thus $t := \frac{\xi}{20} y \geq 7\mu_v$ holds, then Lemma 12(ii) immediately implies the second claim. \square

When we use Lemma 20, we will apply it to all $O(n)$ vertices in each of the $O(\log n)$ iterations of the overall algorithm. So in order for a union bound to give something meaningful, we need a success probability of $1 - \Omega(1/(n \log n))$. We will therefore set $y = \Theta(\xi^{-1} \log n) = \Theta(\varepsilon^{-2} \log^3 n \log \log n)$.

We can sample a uniformly random set $Y \subseteq N(X)$, using the following straightforward procedure. It is the only component of our algorithm that uses independence queries.

Algorithm *SampleNeighbours*: *The algorithm takes as input a set $X \subseteq V$ and an integer y , and it returns a set $Y \subseteq U$ such that $|N(X)| < y$ implies $Y = N(X)$ and $|N(X)| \geq y$ implies that Y is a uniformly random size- y subset of $N(X)$.*

- 1 Let $u_1, \dots, u_{|U|}$ be a uniformly random ordering of U and let $Y = \emptyset$.
- 2 While $|Y| < y$:
 - a Find the smallest i with $u_i \in N(X) \setminus Y$. To do so, we use independence queries of the form $\text{ind}_G(X \cup \{u_1, \dots, u_j\} \setminus Y)$ and perform binary search over $j \in \{1, \dots, |U|\}$.
 - b If u_i was found, add it to Y . Otherwise we have $Y = N(X)$ and return Y .
- 3 Return Y .

LEMMA 21. *The algorithm *SampleNeighbours* is correct, runs in time $O(n \log n)$, and makes at most $O(y \log n)$ independence queries.*

PROOF. The uniform ordering of U induces a uniform ordering of $N(X)$, which implies that `SampleNeighbours` is correct. For the running time, note that Step 1 runs in time $O(n)$ (using Fisher–Yates shuffling) and each binary search runs in time $O(\log n)$. Thus the overall running time is $O(n + y \log n) = O(n \log n)$ and the number of independence queries is $O(y \log n)$. \square

We use `SampleNeighbours` for two purposes: If it returns a set Y of size less than y , then $Y = N(X)$ holds and Y is small enough to compute $\partial(X)$ using the adjacency oracle for all pairs in $Y \times X$. Otherwise the set Y gives us good estimates for the relative degrees of vertices in X by Lemma 20. In particular, we shall use this to approximate the set of vertices in X of high relative degree, as encapsulated by the following definition.

DEFINITION 22. Let $\xi \in \mathbb{R}$ with $0 < \xi \leq 1$ and let $X \subseteq V$. We say $S \subseteq X$ is a ξ -core of X if it satisfies the following properties:

- (W1) every vertex in X with degree at least $\frac{\xi}{8} \cdot |N(X)|$ is contained in S ;
- (W2) every vertex in S has degree at least $\frac{\xi}{32} \cdot |N(X)|$.

We will show in the proof of Theorem 1 that the estimates given by Lemma 20 do indeed yield cores. We now relate cores to balancedness.

LEMMA 23. Let $\xi \in \mathbb{R}$ with $0 < \xi \leq 1$ and let S be a ξ -core of a set $X \subseteq V$.

- (i) If $|S| \geq 32/\xi^2$, then X is ξ -balanced.
- (ii) If $X \setminus S$ contains a vertex of degree at least $\frac{\xi}{4} \cdot |N(X \setminus S)|$, then $|N(X \setminus S)| \leq \frac{1}{2} \cdot |N(X)|$.
Otherwise, $X \setminus S$ is $\frac{\xi}{4}$ -balanced.

PROOF. For the first claim, suppose $|S| \geq 32/\xi^2$. Then by (W2), at least $32/\xi^2$ vertices in X have degree at least $\frac{\xi}{32} \cdot |N(X)|$. Hence $\partial(X) \geq |N(X)|/\xi$ holds, and every vertex $v \in X$ satisfies $d(v) \leq |N(X)| \leq \xi \partial(X)$. Thus X is ξ -balanced.

For the second claim, suppose $v \in X \setminus S$ is a vertex whose degree satisfies $d(v) \geq \frac{\xi}{4} \cdot |N(X \setminus S)|$. Since $v \notin S$, we also have $d(v) \leq \frac{\xi}{8} \cdot |N(X)|$ by (W1). Together, these facts imply $|N(X \setminus S)| \leq \frac{4}{\xi} \cdot d(v) \leq \frac{1}{2} \cdot |N(X)|$ as required. Finally, note that $|N(X \setminus S)| \leq \partial(X \setminus S)$ holds, so if all vertices in $X \setminus S$ have degree at most $\frac{\xi}{4} \cdot |N(X \setminus S)|$, then $X \setminus S$ is $\frac{\xi}{4}$ -balanced by definition. \square

4.3 The Overall Algorithm

Throughout this section, we will take

$$y = \frac{\varepsilon}{800 \log n}, \quad \xi = \frac{\gamma^2}{5 \log \log n} = \frac{\varepsilon^2}{8 \cdot 10^5 \log^2 n \log \log n}, \text{ and}$$

$$y = \frac{4000 \log n}{\xi} = \frac{32 \cdot 10^8 \log^3 n \log \log n}{\varepsilon^2}.$$

The edge counting algorithm works in $O(\log n)$ iterations, starting with $X = V$. In each iteration, either $|X|$ is roughly halved, or $|N(X)|$ is at least halved. We formulate the algorithm recursively.

Algorithm `EdgeCount(X)`: This recursive algorithm takes as input a set $X \subseteq V$ and returns an ε -approximation to $\partial(X)$ with suitably high probability. (Recall that the input graph $G = (U, V, E)$ and the allowed error $\varepsilon > 0$ have already been defined globally.)

- 1 Use `SampleNeighbours(X, y)` to sample a uniformly random $Y \subseteq N(X)$ of size $\min\{y, |N(X)|\}$.
- 2 If $|X| \leq 24 \log n$ or $|Y| < y$, then compute $\partial(X)$ using adjacency queries on $U \times X$ or $Y \times X$, respectively. (if $|Y| < y$, then $Y = N(X)$ holds by the properties of `SampleNeighbours`)

- 3 For all $v \in X$, compute $\tilde{\delta}(v) = \frac{|N(v) \cap Y|}{|Y|}$ using adjacency queries on $Y \times X$.
(w.h.p. each $\tilde{\delta}(v)$ is a $\frac{1}{2}$ -approximation to $\delta(v)$ if $\delta(v) \geq \xi/140$)
- 4 Let $S = \{v \in X : \tilde{\delta}(v) \geq \frac{\xi}{16}\}$. (w.h.p. this is a ξ -core)
- 5 If $\tilde{\delta}(v) \leq \frac{1}{2}\xi$ holds for all $v \in X$, or if $|S| \geq 32/\xi^2$ holds: (w.h.p. X is now ξ -balanced)
 - a Let X' be a uniformly random subset of X . (w.h.p. X' is at most $\frac{3}{4}$ the size of X)
 - b Recursively compute $2 \cdot \text{EdgeCount}(X')$, and return this number.
- 6 Otherwise, independently and uniformly sample $3|U| \log n/\gamma^2$ pairs from $U \times S$, and use the adjacency oracle to determine the number Z of these pairs which are edges in G . Let $\tilde{\partial}(S) := Z\gamma^2|S|/3 \log n$. (w.h.p. $\tilde{\partial}(S) \in (1 \pm \gamma)\partial(S)$.)
- 7 Return $\text{EdgeCount}(X \setminus S) + \tilde{\partial}(S)$. (w.h.p. either $N(X \setminus S)$ is half the size of $N(X)$, or $X \setminus S$ is $\xi/4$ -balanced.)

We are ready to formally prove our main result.

THEOREM 1 (RESTATED). *There is a randomised algorithm \mathcal{A} which, given a rational number ε with $0 < \varepsilon < 1$ and oracle access to an n -vertex bipartite graph G , outputs an ε -approximation of $|E(G)|$ with probability at least $2/3$. Moreover, \mathcal{A} runs in time $\varepsilon^{-2} \cdot O(n \log^4 n \log \log n)$ and makes at most $\varepsilon^{-2} \cdot O(\log^5 n \log \log n)$ calls to the independence oracle.*

PROOF. We may assume without loss of generality that $n \geq 10^5$; otherwise, we simply solve the problem in $O(1)$ time by brute force using the adjacency oracle. Note that each iteration of EdgeCount makes at most one recursive call, so its recursion tree is a path. An *iteration* is an execution of EdgeCount up to a recursive call. We first make a minor modification to EdgeCount : adding a global counter to ensure that we perform at most $t = \lfloor 100 \log n \rfloor$ iterations, otherwise halting with an output of TIMEOUT . We are very unlikely to reach this depth, but this modification will allow us to bound the running time deterministically (as required by Theorem 1). Having done so, we claim that running EdgeCount on input V has the claimed properties.

We first bound the running time for each iteration. By Lemma 21, Step 1 runs in time $O(n \log n)$ and makes at most $O(y \log n)$ independence queries; this step is the only one that makes independence queries at all. Step 2 takes time at most $O(n \log n)$ if $|X| \leq 24 \log n$ or time $O(yn)$ otherwise. Likewise, not counting the recursive calls, Step 3, Step 4, and Step 5 take time $O(yn)$, and Step 6 and Step 7 take time $O(n \log n/\gamma^2) = \varepsilon^{-2}O(n \log^3 n)$. There are $O(\log n)$ total iterations, and $y = \varepsilon^{-2}\Theta(\log^3 n \log \log n)$, so the overall worst-case running time of the algorithm on input V is $O(yn \log n) = \varepsilon^{-2}O(n \log^4 n \log \log n)$, and it makes at most $O(y \log^2 n) = \varepsilon^{-2}O(\log^5 n \log \log n)$ queries to the independence oracle.

Next, we argue that the success probability is at least $2/3$. To reason about this, we define the following events at each recursion depth $1 \leq i \leq t$ of the algorithm:

- $\mathcal{F}_1(i)$ Either Step 3 is not executed at depth i , or each $\tilde{\delta}(v)$ computed indeed either $\frac{1}{2}$ -approximates $\delta(v)$ (if $\delta(v) \geq \xi/140$) or satisfies $\tilde{\delta}(v) \leq \xi/20$ (otherwise).
- $\mathcal{F}_2(i)$ Either Step 5a is not executed at depth i , or $|X'| \leq \frac{3}{4}|X|$ holds and the number $2\partial(X')$ is a 2γ -approximation of $\partial(X)$.
- $\mathcal{F}_3(i)$ Either Step 6 is not executed at depth i , or $\tilde{\partial}(S)$ is a γ -approximation to $\partial(S)$.

Thus $\mathcal{F}_1(i)$, $\mathcal{F}_2(i)$ and $\mathcal{F}_3(i)$ vacuously occur if the algorithm terminates before reaching depth i . We write $\mathcal{F}(i) = \mathcal{F}_1(i) \cap \mathcal{F}_2(i) \cap \mathcal{F}_3(i)$, and $\mathcal{F} = \bigcap_{i=1}^t \mathcal{F}(i)$. We will now show that $\Pr(\mathcal{F}) \geq 2/3$.

Each time Step 3 is executed, the set Y returned by SampleNeighbours in Step 1 has size $y = |Y| \leq |N(X)|$, and thus this set is a uniformly random size- y subset of $N(X)$. Lemma 20 applies and shows that each event $\mathcal{F}_1(i)$ fails to occur for an individual v with probability at most

$\exp(-\xi y/2000)$. By our choice of y , this is precisely $1/n^2$. Since there are at most n vertices v ,

$$\Pr(\mathcal{F}_1(i) \text{ fails}) \leq 1/n. \quad (6)$$

Conditioned on $\mathcal{F}_1(i)$, we claim that the set S defined in Step 4 is a ξ -core. If $\delta(v) \geq \xi/8$, then $\tilde{\delta}(v)$ is a valid $\frac{1}{2}$ -approximation to $\delta(v)$, so $\tilde{\delta}(v) \geq \xi/16$ and thus v is added to S ; this implies that (W1) holds. Conversely, if $\delta(v) < \xi/32$, then either $\tilde{\delta}(v)$ is a $\frac{1}{2}$ -approximation of $\delta(v)$ (in which case $\tilde{\delta}(v) < \xi/16$ and thus v is not added to S) or $\tilde{\delta}(v) \leq \xi/20$ (in which case again v is not added to S); this implies that (W2) holds.

We now claim that if Step 5a is executed, again conditioned on $\mathcal{F}_1(i)$, then X is ξ -balanced. Suppose Step 5a is executed; therefore either $\delta(v) \leq \frac{1}{2}\xi$ holds for all $v \in X$ or $|S| \geq 32/\xi^2$. If $|S| \geq 32/\xi^2$, then X is ξ -balanced by Lemma 23(i), so suppose $\tilde{\delta}(v) \leq \frac{1}{2}\xi$ for all $v \in X$. Since $\mathcal{F}_1(i)$ occurs, for all $v \in X$, either $\tilde{\delta}(v)$ is a $\frac{1}{2}$ -approximation for $\delta(v)$ or $\delta(v) < \xi/140$. In the former case, $\delta(v) \leq 2\tilde{\delta}(v) \leq \xi$, so $\delta(v) \leq \xi$ in both cases and so X is ξ -balanced as claimed.

It follows that conditioned on $\mathcal{F}_1(i)$, each time Step 5a is executed, $|X| \geq 24 \log n$ and X is ξ -balanced. Thus Lemma 19(i) and (ii) apply, so $\mathcal{F}_2(i)$ fails with probability at most $2 \exp(-|X|/24) + 2 \exp(-2\gamma^2/\xi)$. By our choice of ξ , it follows that

$$\Pr(\mathcal{F}_2(i) \text{ fails} \mid \mathcal{F}_1(i)) \leq \frac{2}{n} + \frac{2}{\log^{10} n}. \quad (7)$$

Finally, conditioned on $\mathcal{F}_1(i)$, each time Step 6 is executed, Z is a binomial variable with mean $\mu = 3\partial(S) \log n/\gamma^2|S|$. It follows by Lemma 12(i) that for all i ,

$$\begin{aligned} \Pr(\mathcal{F}_3(i) \text{ fails} \mid \mathcal{F}_1(i)) &= \Pr(|\tilde{\partial}(S) - \partial(S)| > \gamma\partial(S)) = \Pr(|Z - \mu| > \gamma\mu) \\ &\leq 2e^{-\gamma^2\mu/3} = 2e^{-\partial(S) \log n/|S|}. \end{aligned}$$

Since $\mathcal{F}_1(i)$ occurs, S is a ξ -core (as shown above); thus by (W2), every vertex in S has positive degree, and in particular $\partial(S) \geq |S|$. Thus conditioned on $\mathcal{F}_1(i)$, $\mathcal{F}_3(i)$ fails with probability at most $2/n$. In conjunction with (6) and (7), this implies

$$\Pr(\mathcal{F}(i) \text{ fails}) \leq \frac{5}{n} + \frac{2}{\log^{10} n}.$$

Since $n \geq 10^5$ and $t \leq 100 \log n$, this is at most $1/3t$. It follows by a union bound over all $1 \leq i \leq t$ that \mathcal{F} occurs with probability at least $2/3$, as claimed.

Let us now show that conditioned on \mathcal{F} , we do not output TIMEOUT. We claim that in every other iteration, we multiply either $|N(X)|$ or $|X|$ by a factor of at most $\frac{3}{4}$. Since $\mathcal{F}_2(i)$ occurs for all i , it is clear that $|X|$ is multiplied by a factor of at most $\frac{3}{4}$ if the algorithm recurses in Step 5b. If the algorithm recurses in Step 7, then by Lemma 23(ii), either we reduce $|N(X)|$ by at least half, or the set $X \setminus S$ is $\xi/4$ -balanced. In the first case we are done, in the second case it may be that $X \setminus S$ is not significantly smaller than X . However, as $X \setminus S$ is $\xi/4$ -balanced, the condition $\tilde{\delta}(v) \leq \xi/2$ is met for all $v \in X \setminus S$ in the very next iteration of the algorithm (where the input is $X \setminus S$), and then $X \setminus S$ is multiplied by a factor of at most $\frac{3}{4}$. Since initially we have $|X| \leq n$ and $|N(X)| \leq n$, the number of iterations is thus at most $4 \log_{\frac{3}{4}} n < t$ as required.

It remains to prove that conditioned on \mathcal{F} , the function call $\text{EdgeCount}(V)$ returns an ε -approximation for $|E(G)| = \partial(V)$. Let $t' \leq t$ be the total number of iterations; we will prove inductively that for all $0 \leq i \leq t' - 1$, we have $\text{EdgeCount}(X_{t'-i}) \in (1 \pm 2\gamma)^i \partial(X_{t'-i})$. In the last iteration, the algorithm computes $\partial(X_{t'})$ exactly, so the claim is immediate for $i = 0$. If the algorithm in iteration

$t' - i$ recurses in Step 5b, then since $\mathcal{F}_2(t' - i)$ occurs, we have

$$\text{EdgeCount}(X_{t'-i}) = 2 \cdot \text{EdgeCount}(X_{t'-i+1}) \in (1 \pm 2\gamma)^{i-1} \partial(X_{t'-i+1}) \subseteq (1 \pm 2\gamma)^i \partial(X_{t'-i}),$$

as required. If instead it recurses in Step 7, then since $\mathcal{F}_3(t' - i)$ occurs, we have

$$\begin{aligned} \text{EdgeCount}(X_{t'-i}) &= \text{EdgeCount}(X_{t'-i+1}) + \tilde{\partial}(S) \\ &\in (1 \pm 2\gamma)^{i-1} \partial(X_{t'-i+1}) + (1 \pm \gamma) \partial(S) \\ &\subseteq (1 \pm 2\gamma)^i (\partial(X_{t'-i+1}) + \partial(X_{t'-i+1} \setminus X_{t'-i})) = (1 \pm 2\gamma)^i \partial(X_{t'-i}). \end{aligned}$$

Thus the claim holds, and in particular

$$\text{EdgeCount}(V) = \text{EdgeCount}(X_1) \in (1 \pm 2\gamma)^{t-1} \partial(X_1) \subseteq (1 \pm 2\gamma)^t e(G).$$

Since $(1 - 2\gamma)^t \geq 1 - 2t\gamma$ and $(1 + 2\gamma)^t \leq e^{2\gamma t} \leq 1 + 8t\gamma$, it follows that $\text{EdgeCount}(V)$ is a $8t\gamma$ -approximation of $|E(G)|$. Since $t \leq 100 \log n$, by our choice of γ , this is an ε -approximation. \square

5 APPLICATIONS FOR POLYNOMIAL-TIME PROBLEMS

5.1 3SUM

We formally define the problems as follows.

Problem 3SUM expects as input: Three lists A, B and C of integers.

Task: Decide whether there exists a tuple $(a, b, c) \in A \times B \times C$ such that $a + b = c$.

Problem #3SUM expects as input: Three lists A, B and C of integers.

Task: Count the number of tuples $(a, b, c) \in A \times B \times C$ such that $a + b = c$.

THEOREM 4 (RESTATED). *If 3SUM with n integers has a randomised algorithm that runs in time $T(n)$, then there is a randomised ε -approximation algorithm for #3SUM that runs in time $T(n) \cdot \varepsilon^{-2} O(\log^6 n \log \log n)$.*

PROOF. First we note that any bounded-error randomised algorithm for 3SUM must read a constant proportion of the entries in A, B and C , so we can assume $T(n) = \Omega(n)$.

Let (A, B, C) be an instance of #3SUM and let $0 < \varepsilon < 1$. If $\varepsilon \leq \frac{1}{n}$, then we use exhaustive search to solve the problem exactly in time $O(n^3) = O(\varepsilon^{-2} T(n))$. In the following, we assume $\varepsilon > \frac{1}{n}$. Let $E = \{(a, b) \in A \times B : a + b \in C\}$, and let $G = (A, B, E)$. We will proceed by sorting the set C in $O(n \log n)$ time, then applying the algorithm of Theorem 1 to G and ε .

We can evaluate $\text{adj}_G(a, b)$ in time $O(\log n)$ using binary search on C . Moreover, for all $X \subseteq A \cup B$, we have $\text{ind}_G(X) = 1$ if and only if $(X \cap A, X \cap B, C)$ is a ‘no’ instance of 3SUM, so ind_G can be evaluated by solving a single instance of 3SUM, which takes $O(n)$ time to prepare. As in the proof of Theorem 13, we solve the instance by invoking the assumed randomised decision algorithm $100 \log n$ times and outputting the majority answer. The overall algorithm is given by Theorem 1. As this algorithm makes at most $\varepsilon^{-2} \cdot O(\log^6 n) \leq O(n^2 \log^6 n)$ queries to ind_G , the probability that at least one of them is answered incorrectly by the boosted randomised procedure remains negligible, at most $O(1/n)$ by Lemma 12(i), which is in particular at most $1/3$ as required. The overall running time is:

$$\underbrace{O(n \log n)}_{\text{sort } C} + \underbrace{\varepsilon^{-2} O(n \log^4 n \log \log n)}_{\# \text{ queries to } \text{adj}_G} \cdot \underbrace{O(\log n)}_{\text{binary search}} + \underbrace{\varepsilon^{-2} O(\log^5 n \log \log n)}_{\# \text{ queries to } \text{ind}_G} \cdot \underbrace{(O(n) + T(n) \log n)}_{\text{prepare and solve query}}.$$

We have constructed an ε -approximation algorithm for 3SUM that has the claimed running time. \square

THEOREM 5 (RESTATED). *For all $\delta > 0$, there is a randomised ε -approximation algorithm with running time $\varepsilon^{-2} \cdot \tilde{O}(n^{2-\delta/7})$ for instances of #3SUM with n integers such that at least one of A , B , or C may be covered by $n^{1-\delta}$ intervals of length n .*

PROOF. Say a set $S \subseteq \mathbb{Z}$ is (n, δ) -clustered if it can be covered by at most $n^{1-\delta}$ intervals of length n ; note that it can be checked in quasilinear time whether a set is (n, δ) -clustered. Let (A, B, C) be an instance of #3SUM in which at least one of A , B or C is (n, δ) -clustered. By negating and permuting sets if necessary, we may assume that C is (n, δ) -clustered. Exactly as in the proof of Theorem 4, any randomised $T(n)$ -time algorithm for 3SUM on such instances yields a $T(n) \cdot \varepsilon^{-2} O(\log^6 n \log \log n)$ -time randomised approximation scheme. (In particular, note that $(X \cap A, X \cap B, C)$ remains an instance of the restricted problem.) Chan and Lewenstein [8, Corollary 4.3] provide a randomised $O(n^{2-\delta/7})$ -time algorithm for 3SUM on such instances, so the result follows. \square

5.2 Orthogonal Vectors

We formally define the problems as follows.

Problem OV expects as input: Two lists A and B of zero-one vectors in \mathbb{R}^d .

Task: Decide whether there exists a pair $(\mathbf{u}, \mathbf{v}) \in A \times B$ such that $\sum_{i=1}^d \mathbf{u}_i \mathbf{v}_i = 0$.

Problem #OV expects as input: Two lists A and B of zero-one vectors in \mathbb{R}^d .

Task: Count the number of pairs $(\mathbf{u}, \mathbf{v}) \in A \times B$ such that $\sum_{i=1}^d \mathbf{u}_i \mathbf{v}_i = 0$.

THEOREM 2 (RESTATED). *If OV with n vectors in d dimensions has a randomised algorithm that runs in time $T(n, d)$, then there is a randomised ε -approximation algorithm for #OV that runs in time $T(n, d) \cdot \varepsilon^{-2} O(\log^6 n \log \log n)$.*

PROOF. Let (A, B) be an instance of #OV and let $0 < \varepsilon < 1$. If $\varepsilon \leq n^{-2}$ then we can solve the problem exactly in time $O(n^2) = O(\varepsilon^{-1})$, so suppose $\varepsilon > n^{-2}$. Let $E = \{(a, b) \in A \times B : \langle a, b \rangle = 0\}$, and let $G = (A, B, E)$ be a bipartite graph. We will proceed by applying the algorithm of Theorem 1 to G and ε .

We can evaluate adj_G in $O(d)$ time by calculating the inner product. Moreover, for all $X \subseteq A \cup B$, $\text{ind}_G(X) = 1$ if and only if $(A \cap X, B \cap X)$ is a ‘no’ instance of OV, so ind_G can be evaluated by solving a single instance of OV which takes $O(nd)$ time to prepare. As in the proof of Theorem 4, we do so by invoking our randomised decision algorithm $100 \log n$ times and outputting the majority answer. Our overall running time is then

$$\varepsilon^{-2} \cdot O(n \log^4 n \log \log n) \cdot O(d) + \varepsilon^{-2} \cdot (O(nd) + T(n, d) \log n) \cdot O(\log^5 n \log \log n).$$

Since any randomised algorithm for OV must examine a constant proportion of the coordinates of vectors in A and B , we have $T(n, d) = \Omega(nd)$, so the result follows. \square

In the following definitions, \mathcal{R} is a constant finite ring.

Problem OV(\mathcal{R}) expects as input: Two lists A and B of vectors in \mathcal{R}^d .

Task: Decide whether there exists a pair $(\mathbf{u}, \mathbf{v}) \in A \times B$ such that $\sum_{i=1}^d \mathbf{u}_i \mathbf{v}_i = 0_{\mathcal{R}}$.

Problem #OV(\mathcal{R}) expects as input: Two lists A and B of vectors in \mathcal{R}^d .

Task: Count the number of pairs $(\mathbf{u}, \mathbf{v}) \in A \times B$ such that $\sum_{i=1}^d \mathbf{u}_i \mathbf{v}_i = 0_{\mathcal{R}}$.

THEOREM 3 (RESTATED). *Let $m = p^k$ be a constant prime power. There is a randomised ε -approximation algorithm for #OV over $\text{GF}(m)^d$ with running time $\varepsilon^{-2} d^{(p-1)k} \cdot \tilde{O}(n)$, and for #OV over $(\mathbb{Z}/m\mathbb{Z})^d$ with running time $\varepsilon^{-2} d^{m-1} \cdot \tilde{O}(n)$.*

PROOF. Exactly as in the proof of Theorem 2, any randomised $T(n, d)$ -time algorithm for $\text{OV}(\mathcal{R})$ yields a $T(n, d) \cdot \varepsilon^{-2} O(\log^6 n \log \log n)$ -time randomised approximation scheme for #OV(\mathcal{R}). (Note that \mathcal{R} is finite and part of the problem specification, so arithmetic operations require only $O(1)$ time.) The result therefore follows from Theorems 1.6 and 1.3 (respectively) of Williams and Yu [39]. \square

5.3 Negative-Weight Triangles

We formally define the problems as follows.

Problem NWT expects as input: A tripartite graph G and a symmetric function $w : V(G)^2 \rightarrow \mathbb{Z}$.

Task: Decide whether there exists a triangle abc in G such that $w(a, b) + w(b, c) + w(c, a) < 0$.

Problem #NWT expects as input: A tripartite graph G and a symmetric function $w : V(G)^2 \rightarrow \mathbb{Z}$.

Task: Count the number of triangles abc in G such that $w(a, b) + w(b, c) + w(c, a) < 0$.

THEOREM 6 (RESTATED). *If NWT for n -vertex graphs has a randomised algorithm that runs in time $T(n)$, then there is a randomised ε -approximation algorithm for #NWT that runs in time*

$$T(n) \cdot \varepsilon^{-2} O(\log^6 n \log \log n).$$

PROOF. Let (G, w) be an instance of #NWT, let A, B and C be the vertex classes of G , and let $0 < \varepsilon < 1$. If $\varepsilon \leq n^{-3}$ then we can solve the problem exactly in time $O(n^3) = O(\varepsilon^{-1})$, so suppose $\varepsilon > n^{-3}$. Let $U = A$, let $V = \{e \in E(G) : e \subseteq B \cup C\}$, and let

$$E = \left\{ (a, \{b, c\}) \in U \times V : \{a, b\}, \{a, c\} \in E(G) \text{ and } w(a, b) + w(b, c) + w(c, a) < 0 \right\}.$$

Let $H = (U, V, E)$, so that H is a bipartite graph. We will proceed by applying the algorithm of Theorem 1 to H and ε .

We can evaluate adj_H in $O(1)$ time by summing the appropriate weights. Moreover, for all $X \subseteq U \cup V$, define a graph G_X by $V(G_X) = (X \cap A) \cup B \cup C$ and

$$E(G_X) = \left\{ e \in E(G) : e \cap X \cap A \neq \emptyset \text{ or } e \in X \cap V \right\}.$$

Let $w_X = w|_{V(G_X)^2}$. Then for all $X \subseteq U \cup V$, $\text{ind}_H(X) = 1$ if and only if (G_X, w_X) is a ‘no’ instance of NWT, so ind_G can be evaluated by solving a single instance of NWT which takes $O(n^2)$ time to prepare. As in the proof of Theorem 4, we do so by invoking our randomised decision algorithm 100 $\log n$ times and outputting the majority answer. Our overall running time is then

$$\varepsilon^{-2} \cdot O(n^2 \log^4 n \log \log n) \cdot O(1) + \varepsilon^{-2} \cdot (O(n^2) + T(n) \log n) \cdot O(\log^5 n \log \log n).$$

If G is a complete tripartite graph, then any randomised algorithm for NWT must examine a constant proportion of the edges of G , so we have $T(n) = \Omega(n^2)$ and the result follows. \square

In order to approximate algorithm for #NWT, we will reduce to APSP and apply the algorithm of Williams [37]. We formally define APSP as follows.

Problem APSP expects as input: A directed graph G and a function $w : E(G) \rightarrow \mathbb{Z}$ such that G contains no negative-weight cycles under w .

Task: Output the matrix A such that for all $u, v \in V(G)$, $A_{u,v}$ is the minimum weight of any path from u to v in G .

THEOREM 7 (RESTATEd). *There is a randomised ε -approximation algorithm for #NWT which runs in time $\varepsilon^{-2}n^3/e^{\Omega(\sqrt{\log n})}$ on graphs with n vertices and polynomially bounded edge-weights.*

PROOF. By Williams [37, Theorem 1.1], an n -vertex instance of APSP with polynomially bounded edge weights can be solved in time $n^3/e^{\Omega(\sqrt{\log n})}$. There is a well-known reduction from NWT to APSP with only constant overhead, which we give explicitly in the following paragraph. Theorem 6 then implies the existence of an ε -approximation algorithm for #NWT with running time $\varepsilon^{-2}n^3/e^{\Omega(\sqrt{\log n})}$, noting that the polylogarithmic overhead is subsumed into the $e^{\Omega(\sqrt{\log n})}$ term.

It remains only to reduce NWT to APSP. Let (G, w) be an instance of NWT, writing $G = (V, E)$. Form an instance (G', w') of APSP as follows. Let $V(G') = (V \times [3])$, and let

$$E(G') = \bigcup_{i \in \{1,2\}} \bigcup_{\{u,v\} \in E} \{((u, i), (v, i + 1)), ((v, i), (u, i + 1))\}.$$

Let $w'(\{(u, i), (v, i + 1)\}) = w(u, v)$ for all $\{(u, i), (v, i + 1)\} \in E(G')$. Thus for all $\{u, v\} \in E$, each path $(u, 1)(w, 2)(v, 3)$ from $(u, 1)$ to $(v, 3)$ in G' corresponds exactly to the triangle uvw in G , and uvw 's weight is the length of the corresponding path plus $w(u, v)$. Let A be the output of APSP on G' . Then from the discussion above, (G, w) is a 'yes' instance of NWT if and only if for some $\{u, v\} \in E(G)$, we have $A_{(u,1),(v,3)} + w(u, v) < 0$. This can be checked in $O(n^2)$ time. \square

ACKNOWLEDGMENTS

We thank Rahul Santhanam and Ryan Williams for some valuable discussions.

Part of this work was done while the authors were visiting the Simons Institute for the Theory of Computing. The research leading to these results has received funding from the European Research Council (ERC) 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.

REFERENCES

- [1] Amir Abboud, Richard Ryan Williams, and Huacheng Yu. 2015. 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*, Piotr Indyk (Ed.). SIAM, 218–230. <https://doi.org/10.1137/1.9781611973730.17>
- [2] Ilya Baran, Erik D. Demaine, and Mihai Patrascu. 2008. Subquadratic Algorithms for 3SUM. *Algorithmica* 50, 4 (2008), 584–596. <https://doi.org/10.1007/s00453-007-9036-3>
- [3] Paul Beame, Sarel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. 2018. Edge Estimation with Independent Set Oracles. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA (LIPIcs, Vol. 94)*, Anna R. Karlin (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 38:1–38:21. <https://doi.org/10.4230/LIPIcs.ITCS.2018.38>

- [4] Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. 2018. Triangle Estimation using Polylogarithmic Queries. *CoRR* abs/1808.00691 (2018). arXiv:1808.00691 <http://arxiv.org/abs/1808.00691>
- [5] Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. 2019. Hyperedge Estimation using Polylogarithmic Subset Queries. *CoRR* abs/1908.04196 (2019). arXiv:1908.04196 <http://arxiv.org/abs/1908.04196>
- [6] Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. 2018. 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 (LIPIcs, Vol. 123)*, Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 25:1–25:12. <https://doi.org/10.4230/LIPIcs.ISAAC.2018.25>
- [7] Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. 2008. The complexity of Unique k-SAT: An Isolation Lemma for k-CNFs. *J. Comput. Syst. Sci.* 74, 3 (2008), 386–393. <https://doi.org/10.1016/j.jcss.2007.06.015>
- [8] Timothy M. Chan and Moshe Lewenstein. 2015. Clustered Integer 3SUM via Additive Combinatorics. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14–17, 2015*, Rocco A. Servedio and Ronitt Rubinfeld (Eds.). ACM, 31–40. <https://doi.org/10.1145/2746539.2746568>
- [9] Timothy M. Chan and Richard Ryan Williams. 2016. 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*, Robert Krauthgamer (Ed.). SIAM, 1246–1255. <https://doi.org/10.1137/1.9781611974331.ch87>
- [10] Xi Chen, Amit Levi, and Erik Waingarten. 2020. Nearly optimal edge estimation with independent set queries. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5–8, 2020*, Shuchi Chawla (Ed.). SIAM, 2916–2935. <https://doi.org/10.1137/1.9781611975994.177>
- [11] Holger Dell, John Lapinskas, and Kitty Meeks. 2020. 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*, Shuchi Chawla (Ed.). SIAM, 2201–2211. <https://doi.org/10.1137/1.9781611975994.135>
- [12] Martin E. Dyer, Leslie Ann Goldberg, Catherine S. Greenhill, and Mark Jerrum. 2004. The Relative Complexity of Approximate Counting Problems. *Algorithmica* 38, 3 (2004), 471–500. <https://doi.org/10.1007/s00453-003-1073-y>
- [13] Anka Gajentaan and Mark H. Overmars. 1995. On a Class of $O(n^2)$ Problems in Computational Geometry. *Comput. Geom.* 5 (1995), 165–185. [https://doi.org/10.1016/0925-7721\(95\)00022-2](https://doi.org/10.1016/0925-7721(95)00022-2)
- [14] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Richard Ryan Williams. 2017. Completeness for First-Order Properties on Sparse Structures with Algorithmic Applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16–19*, Philip N. Klein (Ed.). SIAM, 2162–2181. <https://doi.org/10.1137/1.9781611974782.141>
- [15] Timon Hertli. 2014. 3-SAT Faster and Simpler - Unique-SAT Bounds for PPSZ Hold in General. *SIAM J. Comput.* 43, 2 (2014), 718–729. <https://doi.org/10.1137/120868177>
- [16] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. 2012. A satisfiability algorithm for AC^0 . In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17–19, 2012*, Yuval Rabani (Ed.). SIAM, 961–972. <https://doi.org/10.1137/1.9781611973099.77>
- [17] Russell Impagliazzo and Ramamohan Paturi. 2001. On the Complexity of k-SAT. *J. Comput. Syst. Sci.* 62, 2 (2001), 367–375. <https://doi.org/10.1006/jcss.2000.1727>
- [18] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.* 63, 4 (2001), 512–530. <https://doi.org/10.1006/jcss.2001.1774>
- [19] Svante Janson, Tomasz Łuczak, and Andrzej Ruciński. 2000. *Random graphs*. Wiley. <https://doi.org/10.1002/9781118032718>
- [20] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. 2004. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM* 51, 4 (2004), 671–697. <https://doi.org/10.1145/1008731.1008738>
- [21] Konstantin Kutsov. 2007. New upper bound for the #3-SAT problem. *Inf. Process. Lett.* 105, 1 (2007), 1–5. <https://doi.org/10.1016/j.ipl.2007.06.017>
- [22] Moritz Müller. 2006. Randomized Approximations of Parameterized Counting Problems. In *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13–15, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4169)*, Hans L. Bodlaender and Michael A. Langston (Eds.). Springer, 50–59. https://doi.org/10.1007/11847250_5
- [23] Mihai Patrascu. 2010. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5–8 June 2010*, Leonard J. Schulman (Ed.). ACM, 603–610. <https://doi.org/10.1145/1806689.1806772>

- [24] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. 2005. An improved exponential-time algorithm for k -SAT. *J. ACM* 52, 3 (2005), 337–364. <https://doi.org/10.1145/1066100.1066101>
- [25] Manuel Schmitt and Rolf Wanka. 2013. Exploiting independent subformulas: A faster approximation scheme for $\#k$ -SAT. *Inf. Process. Lett.* 113, 9 (2013), 337–344. <https://doi.org/10.1016/j.ipl.2013.02.013>
- [26] Johannes Siemons (Ed.). 1989. *Surveys in Combinatorics, 1989*. Cambridge University Press. <https://doi.org/10.1017/cbo9781107359949>
- [27] Michael Sipser. 1983. A Complexity Theoretic Approach to Randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas (Eds.). ACM, 330–335. <https://doi.org/10.1145/800061.808762>
- [28] Larry J. Stockmeyer. 1985. On Approximation Algorithms for $\#P$. *SIAM J. Comput.* 14, 4 (1985), 849–861. <https://doi.org/10.1137/0214060>
- [29] Marc Thurley. 2012. An Approximation Algorithm for $\#k$ -SAT. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France (LIPIcs, Vol. 14)*, Christoph Dürr and Thomas Wilke (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 78–87. <https://doi.org/10.4230/LIPIcs.STACS.2012.78>
- [30] Sinosuke Toda. 1991. PP is as Hard as the Polynomial-Time Hierarchy. *SIAM J. Comput.* 20, 5 (1991), 865–877. <https://doi.org/10.1137/0220053>
- [31] Patrick Traxler. 2016. The Relative Exponential Time Complexity of Approximate Counting Satisfying Assignments. *Algorithmica* 75, 2 (2016), 339–362. <https://doi.org/10.1007/s00453-016-0134-y>
- [32] Leslie G. Valiant. 1979. The Complexity of Computing the Permanent. *Theor. Comput. Sci.* 8 (1979), 189–201. [https://doi.org/10.1016/0304-3975\(79\)90044-6](https://doi.org/10.1016/0304-3975(79)90044-6)
- [33] Leslie G. Valiant and Vijay V. Vazirani. 1986. NP is as Easy as Detecting Unique Solutions. *Theor. Comput. Sci.* 47, 3 (1986), 85–93. [https://doi.org/10.1016/0304-3975\(86\)90135-0](https://doi.org/10.1016/0304-3975(86)90135-0)
- [34] Virginia Vassilevska Williams. 2015. 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 (LIPIcs, Vol. 43)*, Thore Husfeldt and Iyad A. Kanj (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 17–29. <https://doi.org/10.4230/LIPIcs.IPEC.2015.17>
- [35] Virginia Vassilevska Williams and Richard Ryan Williams. 2018. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. *J. ACM* 65, 5 (2018), 27:1–27:38. <https://doi.org/10.1145/3186893>
- [36] Richard Ryan Williams. 2005. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.* 348, 2-3 (2005), 357–365. <https://doi.org/10.1016/j.tcs.2005.09.023>
- [37] Richard Ryan Williams. 2014. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, David B. Shmoys (Ed.). ACM, 664–673. <https://doi.org/10.1145/2591796.2591811>
- [38] Richard Ryan Williams. 2018. Counting Solutions to Polynomial Systems via Reductions. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA (OASICS, Vol. 61)*, Raimund Seidel (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:15. <https://doi.org/10.4230/OASICS.SOSA.2018.6>
- [39] Richard Ryan Williams and Huacheng Yu. 2014. Finding orthogonal vectors in discrete structures. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, Chandra Chekuri (Ed.). SIAM, 1867–1877. <https://doi.org/10.1137/1.9781611973402.135>