Fine-Grained Reductions from Approximate Counting to Decision

Holger Dell
Saarland University
Cluster of Excellence (MMCI)
Saarbrücken, Germany
hdell@mmci.uni-saarland.de

John Lapinskas
University of Oxford
Department of Computer Science
Oxford, UK
John.Lapinskas@cs.ox.ac.uk

ABSTRACT

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 $\mathrm{GF}(m)^d$ for constant m can be solved in time $n \cdot \mathrm{poly}(d)$ 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 $O(c^n)$-time algorithm for k-SAT. Then we prove that for all k, there is an $O((c+o(1))^n)$-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 $1+\varepsilon$ in time $2^{o(n)}/\varepsilon^2$ (taking $\varepsilon > 0$ as part of the input). A full version of this paper containing detailed proofs is available at https://arxiv.org/abs/1707.04609.

STOC'18, June 25–29, 2018, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5559-9/18/06…$15.00
https://doi.org/10.1145/3188745.3188920

CCS CONCEPTS

• **Theory of computation → Sketching and sampling**; *Graph algorithms analysis*; *Parameterized complexity and exact algorithms*; *Probabilistic computation*; *Stochastic approximation*;

KEYWORDS

Approximate counting, fine-grained complexity, orthogonal vectors, satisfiability, isolation lemma

ACM Reference Format:
Holger Dell and John Lapinskas. 2018. Fine-Grained Reductions from Approximate Counting to Decision. In *Proceedings of 50th Annual ACM SIGACT Symposium on the Theory of Computing (STOC'18).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3188745.3188920

1 INTRODUCTION

1.1 Overview

It is clearly at least as hard to count objects as it is to decide their existence, and very often it is harder. For example, Valiant [24] defined the class #P as the natural counting variant of NP. Toda [22] proved that $\mathrm{P}^{\#\mathrm{P}}$ contains the entire polynomial hierarchy (including P^{NP}), so a #P-oracle is more powerful than an NP-oracle unless the polynomial hierarchy collapses. The decision counterparts of many #P-complete problems are in P; for example, counting perfect matchings is #P-complete and detecting one is in P.

However, the situation changes substantially if we consider approximate rather than exact counting. For all real ε with $0 < \varepsilon < 1$, we say that $x \in \mathbb{R}$ is an *ε-approximation* to $N \in \mathbb{R}$ if $|x - N| \leq \varepsilon N$ holds. Clearly, computing an ε-approximation to N (taking ε as part of the input) is at least as hard as deciding whether $N > 0$ holds, but surprisingly, in many settings it is no harder. Indeed, Sipser [19] and Stockmeyer [20] proved implicitly that every problem in #P has a polynomial-time randomised ε-approximation algorithm using an NP-oracle; the result is later proved explicitly in Valiant and Vazirani [25]. This is a foundational result in the wider complexity theory of polynomial approximate counting initiated by Dyer, Goldberg, Greenhill and Jerrum [7].

Another example arises in parameterised complexity. Here, the usual goal is to determine whether an instance of size n with parameter k can be solved in "FPT time" $f(k)\cdot\mathrm{poly}(n)$ for some computable function $f : \mathbb{N} \to \mathbb{N}$. Hardness results are normally presented using the W-hierarchy (see for example [8]). Müller [16] has proved that for any problem in #W[1], there is a randomised ε-approximation algorithm using a W[1]-oracle which runs on size-n parameter-k instances in time $f(k) \cdot \mathrm{poly}(n, \varepsilon^{-1})$ for some computable $f : \mathbb{N} \to \mathbb{N}$. He also proved analogous results for the rest of the hierarchy.

In fine-grained complexity, we are concerned not merely with the classification of algorithms into broad categories such as polynomial, FPT, or subexponential, but with their more precise running times. Alongside SAT, perhaps the most important problems in the field are 3SUM, Orthogonal Vectors (OV), 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 Williams [29] for a recent survey. It is not clear what a "canonical" counting version of APSP should be, so we instead consider the Negative-Weight Triangle problem (NWT), which is equivalent to APSP under subcubic reductions [30].

All known reductions from approximate counting to decision in this setting suffer from a common issue. As a concrete example, we consider a result of Thurley [21]: 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 implies an analogue of the results above; if the Strong Exponential Time Hypothesis (SETH) of Impagliazzo, Paturi and Zane [13] is false, then algorithms not only for k-SAT but also for approximate #k-SAT outperform exhaustive search by an exponential factor as $k \to \infty$. However, the exponential savings go down from δ for decision to $\delta/2$ for counting. From a fine-grained perspective, Thurley's algorithm would be far more interesting if its running time was $\varepsilon^{-2} \cdot 2^{(1-\delta+o(1))n}$, thus guaranteeing that k-SAT and approximate #k-SAT have essentially the same time complexity as $k \to \infty$. Likewise, no reductions from approximate #OV, #3SUM or #NWT to their decision counterparts with subpolynomial overhead are known.

The most important contribution of our paper is a framework for reductions from approximate counting problems to decision problems with an overhead that is only polylogarithmic in the input size. In particular, our framework is sufficiently general to yield such reductions from approximate #OV, #3SUM and #NWT to their decision counterparts without much effort. As an example, consider an n-element instance of #3SUM, in which we are given three lists A, B and C of integers and must output the number of tuples $(a, b, c) \in A \times B \times C$ with $a + b = c$. Assuming for simplicity that the entries of the lists are polynomially bounded in n, a decision algorithm for 3SUM with running time $O(n^{2-\delta(n)})$ for some $\delta(n) > 0$ would yield an ε-approximation algorithm for #3SUM with running time $\varepsilon^{-4} \cdot \tilde{O}(n^{2-\delta(n)})$ under our reduction. Thus the algorithm remains better than brute force even for small values of ε; in fact, the running time improves to $\varepsilon^{-2} \cdot \tilde{O}(n^{2-\delta(n)})$ if $\delta(n) < 1/2$.

OV, 3SUM and NWT all have simple algorithms, which generalise to exact counting and over which it is conjectured that no polynomial improvement is possible. However, we consider it very unlikely that these conjectures will be proved in the near future, and our results are unconditional. Moreover, these conjectures do not rule out superpolylogarithmic improvements to the naive algorithms, and such improvements are already known for OV [1] and NWT [27]. Our results imply that all such improvements immediately translate into approximate counting algorithms with very similar running times; in particular, the algorithm of [27] currently has no counterpart for #NWT, so we already obtain a concrete algorithmic result. Our framework is also general enough to easily accommodate several variants and special cases of OV, 3SUM and NWT for which even polynomial improvements are known, again leading to improved approximate counting algorithms. Thus our results would remain of interest even if the conjectures were proved.

Independently of our framework, we also prove a fine-grained version of Thurley's result as $k \to \infty$: If (randomised)[1] SETH is false and k-SAT may be solved in time $O(2^{(1-\delta)n})$ for all k, then there is an $\varepsilon^{-2} \cdot 2^{(1-\delta+o(1))n}$-time ε-approximation algorithm for #k-SAT. In particular, by the sparsification lemma this implies that the widely-believed Exponential Time Hypothesis (ETH, see [12]) is equivalent to the seemingly-weaker statement that there is no algorithm to ε-approximate #3-SAT in time $\varepsilon^{-2} \cdot 2^{o(n)}$. The question of whether there is a fine-grained reduction from approximate #k-SAT to k-SAT for fixed k remains open.

The remainder of Section 1 consists of a detailed statement of our results and comparison to past work.

1.2 The General Framework

We will use OV as a motivating example to explain our general framework. In this problem, 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 such pairs. Writing $n = |A| + |B|$, it is easy to see that OV and #OV can both be solved in $O(n^2 d)$ operations by iterating over all pairs, and it is conjectured [26] that for all $\delta > 0$, when $d = \omega(\log n)$, OV admits no $O(n^{2-\delta})$-time randomised algorithm. This conjecture is implied by SETH [26], and Abboud, Williams and Yu [1] proved that it fails when $d = O(\log n)$. Normally, it is assumed that d is polylogarithmic in n.

We reduce OV to the problem of approximately counting edges in an arbitrary n-vertex bipartite graph G to which we have only limited oracle access. Our two oracles are an adjacency oracle, which decides whether a given vertex pair is adjacent, and an independence oracle, which decides whether a given vertex set contains any edges. A somewhat informal statement of our main result is as follows.

Theorem 1. *There is a randomised algorithm $\mathcal{A}$ which, given a rational $0 < \varepsilon < 1$ and 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 $\varepsilon^{-4} \cdot \tilde{O}(n)$ and makes $\varepsilon^{-2} \cdot O((\log n)^6)$ calls to the independence oracle.*

We prove this result (and state it in more detail) as Theorem 10 in Section 3. Note that oracle calls are usually modelled as taking constant time, so the adjacency oracle is called at most $\varepsilon^{-4} \cdot \tilde{O}(n)$ times. In the case of OV, the reduction is very simply described; the vertex classes are A and B, and a pair of vertices are adjacent if and only if they are orthogonal. We can simulate an adjacency query naively in time $O(d) = \tilde{O}(1)$, and we can simulate an independence query on a set X by applying a decision algorithm to the subinstance $(X \cap A, X \cap B)$ of OV. Theorem 1 therefore implies the following result.

[1] To streamline our discussion, we ignore the detail that some papers only allow for deterministic algorithms. Throughout the paper, we require randomised algorithms to have success probability at least $2/3$ unless otherwise specified.

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*

$$\varepsilon^{-2}T(n,d)\cdot O((\log n)^6)+\varepsilon^{-4}d\cdot\tilde{O}(n)\,.$$

It is easy to see that there is no $o(n)$-time algorithm for OV, so $T(n,d)=\Omega(n)$. Thus Theorem 2 implies that regardless of the time complexity of OV, when $\varepsilon>0$ is constant, we can ε-approximate #OV with only polylogarithmic overhead over decision. Moreover, the $\varepsilon^{-4}d\cdot\tilde{O}(n)$ term can be dropped unless the Orthogonal Vector Conjecture is very badly wrong and $T(n,d)=O(n^{3/2})$. Indeed, if $T(n,d)=\omega(n^{3/2})$ and $\varepsilon=\Omega(n^{-1/4})$, then the first term dominates the second term. If instead $\varepsilon=O(n^{-1/4})$, then $\varepsilon^{-2}T(n,d)=\Omega(n^2)$ and we can solve #OV exactly by brute force to obtain the required running time.

There is currently no significant gap between the best-known algorithms for OV [1] and #OV (due to Chan and Williams [5]), as both have running time $n^{2-1/O(\log(d/\log n))}$. However, Theorem 2 does imply that any improvement on [1] will immediately translate into an improved approximate counting algorithm. Moreover, a version of OV in which the real zero-one vectors are replaced by arbitrary vectors over finite fields or rings is studied by Williams and Yu [28], who obtain efficient randomised algorithms for this problem. Even though their paper did not consider the counting version and their algorithms do not immediately generalise to counting, we can nevertheless use their decision algorithm as a black box and apply our general framework in Theorem 1 to obtain efficient approximate counting algorithms for this problem.

Theorem 3. *Let $m=p^k$ be a constant prime power. Then there is a randomised ε-approximation algorithm for n-vector instances of* #OV *over* $\mathrm{GF}(m)^d$ *(resp. $(\mathbb{Z}/m\mathbb{Z})^d$) in time $\varepsilon^{-4}d^{(p-1)k}\cdot\tilde{O}(n)$ (resp. $\varepsilon^{-4}d^{m-1}\cdot\tilde{O}(n)$).*

The dependence on d may be close to best possible; under SETH, for all $\delta>0$ and $f:\mathbb{N}\to\mathbb{N}$ with $f(x)=o(x/\log x)$, OV over $\mathrm{GF}(m)^d$ (resp. $(\mathbb{Z}/m\mathbb{Z})^d$) cannot be solved in time $n^{2-\delta}d^{f((p-1)k)}$ (resp. $n^{2-\delta}d^{f(m)}$) for all but finitely many values of $m=p^k$ [28].

1.3 3SUM and NWT

We now give additional applications of Theorem 1 to 3SUM and NWT. The proofs of these results are simple, and we defer them to the full version [6]. Recall that 3SUM asks, given three integer lists A, B and C of total length n, whether there exists a tuple $(a,b,c)\in A\times B\times C$ with $a+b=c$. (Frequently the input is taken to be a single list rather than three; it is well-known that the two versions are equivalent.) One popular extension of this problem is 3SUM+, due to Williams and Williams [30], 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 defined in the previous section. 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 [9, 18] that for all $\delta>0$, 3SUM admits no $O(n^{2-\delta})$-time randomised algorithm. Using Theorem 1, we obtain the following analogue of Theorem 2.

Theorem 4. *If* 3SUM *with n integers from $[-N,N]$ has a randomised algorithm that runs in time $T(n,N)$, then there is a randomised ε-approximation algorithm for* #3SUM *that runs in time*

$$\varepsilon^{-2}T(n,N)\cdot O((\log n)^6)+\varepsilon^{-4}\log N\cdot\tilde{O}(n)\,.$$

There is an easy FFT-based algorithm for both 3SUM and #3SUM with running time $\tilde{O}(N+n)$, so N is normally assumed to be $\Omega(n^2)$. It is also easy to show that $T(n,N)=\Omega(n)$. Thus assuming N is bounded above by a polynomial in n, as is usual, Theorem 4 implies that our algorithm 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 essentially the same time complexity.

The best-known algorithm for 3SUM, due to Baran, Demaine and Pătraşcu [2], has running time $O(n^2(\log\log n/\log n)^2)$. Thus Theorem 4 does not currently yield improved algorithms for #3SUM. However, Chan and Lewenstein [4] have proved that the 3SUM conjecture fails when the problem is restricted to instances in which elements of one list are somewhat clustered, in a sense made explicit below. (In fact, their algorithm also works for 3SUM+.) This is an interesting special case with several applications, including monotone multi-dimensional 3SUM with linearly-bounded coordinates — see the introduction of [4] for an overview. By combining this algorithm with an analogue of Theorem 4, we obtain the following result.

Theorem 5. *For all $\delta>0$, there is a randomised ε-approximation algorithm with running time $\varepsilon^{-4}\cdot\tilde{O}(n^{2-\delta/7})$ for instances of* #3SUM *with n polynomially bounded integers such that at least one of A, B, or C may be covered by $n^{1-\delta}$ intervals of length n.*

Finally, we study NWT, the problem of deciding whether an edge-weighted tripartite graph contains a triangle of negative total weight. Williams and Williams [30] have shown that NWT is equivalent to APSP under subcubic reductions. It is easy to see that NWT can be solved in $\tilde{O}(n^3)$ operations by checking every possible triangle, and it is conjectured [30] that for all $\delta>0$, NWT admits no $O(n^{3-\delta})$-time randomised algorithm. Using our framework in Theorem 1, we obtain the following result for the natural counting version #NWT of NWT, in which we approximate the number of negative-weight triangles.

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

$$\varepsilon^{-2}T(n,W)\cdot O((\log n)^6)+\varepsilon^{-4}\log W\cdot\tilde{O}(n^2)\,.$$

Note that it is impossible to decide NWT in time $o(n^2)$, so when W is polynomially bounded, our algorithm has only polylogarithmic overhead over decision. Note also that it is known [30] that a truly subcubic algorithm for NWT would imply a truly subcubic listing algorithm. While a listing algorithm is obviously stronger than an approximate counting algorithm, this reduction has polynomial overhead and so does not imply Theorem 6. Together with an algorithm of Williams [27], Theorem 6 implies the following.

Theorem 7. *There is a randomised ε-approximation which runs on n-vertex instances of* #NWT *with polynomially bounded weights in time $\varepsilon^{-4}n^3/e^{\Omega(\sqrt{\log n})}$.*

1.4 SAT

Recall that SETH fails if and only if there exists $\delta > 0$ such that for all k, there is a k-SAT algorithm with running time $O(2^{(1-\delta)n})$, where n is the number of variables in the input formula. An immediate corollary of Thurley's reduction [21] is that in this case, for all $k \geq 1$, there is also an $\varepsilon^{-2} \cdot O^*(2^{(1-\delta/2)n})$-time ε-approximation algorithm for k-SAT. However, this reduction has exponential overhead and so is not fine-grained. Traxler [23] provides a reduction with subexponential overhead for a related problem, in which $k = \Omega(\log n)$ holds. We provide a reduction from approximate #k-SAT to k-SAT for constant k, with overhead roughly $2^{((\log k)^2/k)n}$, which we formally state as Theorem 11 in Section 4. (We also sketch the proof.) In particular, letting $k \to \infty$ yields the following easy corollary.

Theorem 8. *Let $\delta > 0$. 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 $\varepsilon^{-2} \cdot O(2^{(1-\delta+\delta')n})$.*

Thus if SETH is false for some $\delta > 0$, then for all $k \in \mathbb{N}$ we obtain an $\varepsilon^{-2} \cdot 2^{(1-\delta+o(1))n}$-time ε-approximation algorithm for #k-SAT. There is no particular reason to believe that an efficient decision algorithm would yield an efficient counting algorithm directly. Indeed, when $k = 3$, the most efficient known algorithms run in time $O(1.308^n)$ for decision (due to Hertli [10]), in time $O(1.537^n)$ for $\frac{1}{2}$-approximate counting (due to Thurley [21]), and in time $O(1.642^n)$ for exact counting (due to Kutzkov [14]).

Recall that ETH holds if and only if there is no subexponential-time algorithm for 3-SAT. By the sparsification lemma of Impagliazzo, Paturi, and Zane [13], a subexponential-time algorithm for 3-SAT would imply subexponential-time algorithms for k-SAT for all k. Thus by letting δ increase to 1 in Theorem 8, we see that ETH is implied by a seemingly-weaker approximate counting formulation.

Theorem 9. *ETH is true if and only if there exists $\delta > 0$ such that no randomised ε-approximation algorithm can run on n-variable instances of* #3-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. For (large) fixed k, the best-known decision, $\frac{1}{2}$-approximate counting, and exact counting algorithms (due to Paturi, Pudlák, Saks, and Zane [17], Thurley [21], and Impagliazzo, Matthews, and Paturi [11], respectively) all have running time $2^{(1-\Theta(1/k))n}$, but with progressively worse constants in the exponent. Thus if the overhead of our reduction could be reduced even to $2^{o(1/k)\cdot n}$, we would obtain improved approximate counting algorithms for fixed k.

2 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 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 of v, that is,

$$N(v) = \{w \in V(G) : \{v, w\} \in E(G)\}.$$

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.

In general, we shall not be overly concerned with optimising logarithmic factors in running times. We shall write $f(x) = \tilde{O}(g(x))$ when for some constant $c \in \mathbb{R}$, we have $f(x) = O((\log x)^c g(x))$ as $x \to \infty$. Similarly, we write $f(x) = O^*(g(x))$ when for some constant $c \in \mathbb{R}$, we have $f(x) = O(x^c g(x))$ as $x \to \infty$.

3 APPROXIMATE COUNTING TO DECISION: A GENERAL FINE-GRAINED REDUCTION

Our main reduction can be viewed as an efficient algorithm in a model of computation where the algorithm observes the graph using query access. We formalize the setting as follows.

Definition 1. *Let $G = (U, V, E)$ be a bipartite graph. We define the* independence oracle *of G to be the function*

$$\mathrm{ind}_G : 2^{U \cup V} \to \{0, 1\},$$

such that $\mathrm{ind}_G(S) = 1$ if and only if S is an independent set in G. We define the adjacency oracle *of G to be the function*

$$\mathrm{adj}_G : U \times V \to \{0, 1\},$$

such that $\mathrm{adj}_G(u, v) = 1$ if and only if $(u, v) \in E$.

In our applications, the edges of G correspond to witnesses of a decision problem. For example, in OV, they correspond to pairs of orthogonal vectors. Thus calling adj_G corresponds to verifying a potential witness, and calling ind_G corresponds to solving the decision problem on a sub-instance. Our main result is the following formal version of Theorem 1.

Theorem 10. *There is a randomised algorithm $\mathcal{A}$ satisfying the following. Suppose that $\mathcal{A}$ is given access to the independence and adjacency oracles of some bipartite graph $G = (U, V, E)$, together with U, V and some rational ε with $0 < \varepsilon < 1$. Then with probability at least $\frac{2}{3}$, algorithm $\mathcal{A}$ returns an ε-approximation to $e(G)$. Moreover, writing $n = |U \cup V|$, algorithm $\mathcal{A}$ runs in time $\varepsilon^{-4} \cdot \tilde{O}(n)$ and calls ind_G at most $\varepsilon^{-2} \cdot O((\log n)^6)$ times.*

Throughout the rest of the section, we take $G = (U, V, E)$ to be the bipartite graph of the theorem statement and $n = |U \cup V|$. Moreover, for all $X \subseteq V$, we define $\partial(X) = |E \cap (U \times X)|$ and $U_X = \{u \in U : (u, v) \in E \text{ for some } v \in X\}$.

We briefly compare the performance of the algorithm of Theorem 10 with that of the standard approach of sampling to deal with dense instances combined with brute force counting to deal with sparse instances (as used in Thurley [21]). Suppose ε is constant, that we can evaluate ind_G in time $O(n^{2-\alpha})$ for some $\alpha > 0$, that we can evaluate adj_G in time $O(1)$, and that the input graph contains n^β edges for some $\beta > 0$. Then sampling requires $\Omega(n^{2-\beta})$ time, and brute force enumeration of the sort used in [21] requires $\Omega(n^{2-\alpha+\beta})$

time. The worst case arises when $\beta = \alpha/2$, in which case the algorithm requires $\Omega(n^{2-\alpha/2})$ time. However, the algorithm of Theorem 10 requires only $\tilde{O}(n^{2-\alpha})$ time in all cases. Thus it has only polylogarithmic overhead over deciding whether the graph contains edges at all.

We now sketch the proof of Theorem 10. For all $X \subseteq V$, we may halve $\partial(X)$ in expectation simply by removing half the vertices in X chosen binomially at random. Moreover, if $\partial(X)$ is sufficiently small, we may use binary search to efficiently determine $\partial(X)$ exactly. Thus we might hope to implement the classical approach of Valiant and Vazirani [25]; 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 approach fails. Indeed, 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 2). If X is not balanced, then after removing half the vertices in X, the new value of $\partial(X)$ will not be concentrated around its expectation. Thus the output of the naive algorithm above will not be concentrated around $e(G)$.

In Lemma 3, we show using martingale inequalities that unbalancedness is the only thing that can go wrong; if X is balanced, then with high probability we can approximately halve $\partial(X)$ by randomly deleting half of X. We therefore proceed by finding a small set of vertices which "unbalances" X, counting the edges incident to them with brute force, removing them from X to make it balanced, then deleting 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.

In Lemma 5, we describe an efficient randomised algorithm (**FindCore**) which, if $\partial(X)$ is too large to determine exactly, finds an approximation to the set of vertices in X whose neighbours are dense in U_X; we call this approximation a *core* (see Definition 4). If the core is empty or large (a *witness*), we show in Lemma 6(i) that X is balanced and we can proceed with halving. If the core is non-empty but small (an *unbalancer*), we will enumerate the edges incident to it by brute force and remove it from X. In Lemma 6(ii), we show that having done so, either X is balanced (with slightly worse parameters) or we have halved $|U_X|$. Thus by repeating this procedure at most $\lg n$ times, we can either make X balanced or make $\partial(X)$ sufficiently small to determine exactly by binary search. (Actually, we are a little more careful than this in order to optimise the running time.) A formal statement of our algorithm is given as **EdgeCount** on p. 6, followed by a correctness proof.

We expand the above sketch into a formal proof of Theorem 10.

Definition 2. *Given $0 < \xi < 1$, we say a non-empty set $X \subseteq V$ is* ξ-balanced *if every vertex in X has degree at most $\xi\partial(X)$.*

Lemma 3. *Let $0 < \xi < 1$, suppose $X \subseteq V$ is ξ-balanced, and suppose $|X| \geq 24 \log n$. Let $X' \subseteq X$ be a random subset formed by including each vertex of X independently with probability $\frac{1}{2}$. Then with probability at least $1 - 4/n$, we have*

$$|X'| \leq 3|X|/4 \quad \text{and} \quad |\partial(X') - \partial(X)/2| \leq \sqrt{\xi \log n} \cdot \partial(X)\,.$$

Proof. First note that since $\mathbb{E}(|X'|) = |X|/2$, by a standard Chernoff bound we have

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

Now let $R = \sqrt{\xi \log n}$. For each vertex $v \in X$, let i_v be the indicator random variable of the event that $v \in X'$. Note that $\partial(X')$ is a function of $\{i_v : v \in X\}$, and that changing a single indicator variable i_v alters $\partial(X')$ by exactly $d(v)$. Moreover, $\mathbb{E}(\partial(X')) = \partial(X)/2$. Hence by McDiarmid's inequality [15],

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

Let $t = |X|$, and define a function $f : \mathbb{R}^t \to \mathbb{R}$ by

$$f(x_1, \ldots, x_t) = \sum_{i=1}^{t} x_i^2\,.$$

Suppose that $(x_1, \ldots, x_t)$ maximises f subject to the constraints $0 \leq x_1, \ldots, x_t \leq \xi\partial(X)$ and $\sum_{i=1}^t x_i = \partial(X)$. For all $0 < c \leq x \leq y$, we have $((x-c)^2 + (y+c)^2) - (x^2 + y^2) = 2c(-x + y + c) > 0$, so we must have $x_i \in \{0, \xi\partial(X)\}$ for all but at most one value of $i \in [t]$. (Otherwise, we could increase the value of f by perturbing $x_1, \ldots, x_t$.) Thus up to reordering, we have

- $x_1 = \cdots = x_{\lfloor 1/\xi \rfloor} = \xi\partial(X)$,
- $x_{\lfloor 1/\xi \rfloor + 1} = (1 - \xi \lfloor 1/\xi \rfloor)\partial(X)$, and
- $x_{\lfloor 1/\xi \rfloor + 2}, \ldots, x_t = 0$.

Since X is ξ-balanced, it follows that

$$\begin{aligned}\sum_{v\in X} d(v)^2 &\leq \left\lfloor \frac{1}{\xi} \right\rfloor \xi^2\partial(X)^2 + \left(1 - \xi\left\lfloor \frac{1}{\xi} \right\rfloor\right)^2 \partial(X)^2 \\ &\leq (\xi + \xi^2)\partial(X)^2 \leq 2\xi\partial(X)^2.\end{aligned}$$

By (2), it follows that

$$\mathbb{P}\left(\left|\partial(X') - \frac{\partial(X)}{2}\right| \geq R\partial(X)\right) \leq 2e^{-R^2/\xi} = \frac{2}{n}. \quad (3)$$

The result therefore follows from (1), (3), and a union bound. □

Definition 4. *Given $0 < \xi < 1$ and $\emptyset \subsetneq X \subseteq V$, 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 $\xi|U_X|$ is contained in S;*
(W2) *every vertex in S has degree at least $\xi|U_X|/24$.*

If $1 \leq |S| < 24/\xi^2$, then we call S a ξ-unbalancer*; on the other hand, if $S = \emptyset$ or $|S| \geq 24/\xi^2$, we call S a* ξ-witness.

To find a ξ-core of X, we use the following procedure. Note that if there are few non-isolated vertices remaining, it will instead return $\partial(X)$ exactly.

Algorithm FindCore(X, ξ)**.** *The input is a set $\emptyset \subsetneq X \subseteq V$ and a rational $0 < \xi < 1$. If $|X| < 24 \log n$ or $|U_X| < 24 \log n/\xi$, it returns $\partial(X)$. Otherwise, it returns a set S which with probability at least $1 - 3/n$ is a ξ-core of X.*

(C1) If $|X| < 24 \log n$, then use adj_G to enumerate the edges incident to X and return the total.
(C2) Let $u_1, \ldots, u_t$ be a uniformly random ordering of U, and let $x = \lceil 24 \log n/\xi \rceil$.

(C3) Recursively define a sequence $k_1, \dots, k_x$ by letting k_i be the maximum number $k \in \{1, \dots, t\}$ that satisfies

$$\mathrm{ind}_G(X \cup \{u_1, \dots, u_k\} \setminus \{u_{k_1+1}, \dots, u_{k_{i-1}+1}\}) = 1\,.$$

Use binary search to compute $k_1, \dots, k_x$. Let Y be the set with

$$Y = \{u_{k_i+1} : i \in [x],\ k_i < t\}\,.$$

(C4) If $k_x = t$, then use adj_G to enumerate the edges incident to Y and return the total.

(C5) Otherwise, use adj_G to find the set $S \subseteq X$ of vertices which are adjacent to at least $\xi x/2$ vertices in Y. Return S.

Lemma 5. *When $n \geq 2$, **FindCore** is correct. Moreover, it runs in $\xi^{-1} \cdot \tilde{O}(n)$ time and makes at most $\xi^{-1} \cdot O((\log n)^2)$ calls to the independence oracle.*

Proof. Let $x = \lceil 24 \log n/\xi \rceil$. We start by analysing the running time: (C1) requires $O(n \log n)$ time and at most $24n \log n$ calls to adj_G; (C2) requires $O(n \log n)$ time; (C3) requires $x \cdot O(\log n)$ time and at most $x \lg n$ calls to ind_G; (C4) and (C5) together require $x \cdot O(n)$ time and at most xn calls to adj_G. Thus the running time guarantees in the statement are correct. It is also immediate that the algorithm behaves correctly if $|X| < 24 \log n$.

Note that when $k_i < t$, we have

$$\begin{aligned} &k_i + 1 \\ &= \min\{k : \mathrm{ind}_G(X \cup \{u_1, \dots, u_k\} \setminus \{u_{k_1+1}, \dots, u_{k_{i-1}+1}\}) = 0\} \\ &= \min\{k \in [t] \setminus \{k_1 + 1, \dots, k_{i-1} + 1\} : u_k \in U_X\}\,. \end{aligned}$$

Thus Y consists of the least x elements of U_X in the random ordering $u_1, \dots, u_t$, or all of U_X if $|U_X| < x$. Thus if $|U_X| < x$ then **FindCore** behaves correctly, and if $|U_X| \geq x$ then Y is a uniformly random size-x subset of U_X. Suppose $|U_X| \geq x$. Then we must prove that S is a ξ-core with probability at least $1 - 3/n$.

Let $Z = \{v \in X \mid d(v) \geq \xi |U_X|\}$. If $v \in Z$, then $|N(v) \cap Y|$ follows a hypergeometric distribution with mean $(d(v)/|U_X|) \cdot x \geq \xi x \geq 24 \log n$. By a standard Chernoff bound, all $v \in Z$ satisfy

$$\mathbb{P}(v \notin S) = \mathbb{P}(|N(v) \cap Y| < \xi x/2) \leq 2e^{-\xi x/12} \leq 2/n^2. \quad (4)$$

Conversely, let $Z' = \{v \in X \mid d(v) < \xi |U_X|/24\}$. For all $v \in Z'$, the random variable $|N(v) \cap Y|$ follows a hypergeometric distribution with mean $(d(v)/|U_X|) \cdot x < \xi x/24$. Again by a standard Chernoff bound, all $v \in Z'$ satisfy

$$\mathbb{P}(v \in S) = \mathbb{P}\Big(|N(v) \cap Y| \geq \xi x/2\Big) \leq e^{-\xi x/2} \leq e^{-12 \log n} < \frac{1}{n^2}. \quad (5)$$

We obtain the required upper bound on the failure probability of the algorithm by (4), (5), and a union bound over all $v \in Z \cup Z'$. □

The following lemma implies that finding a ξ-core S of $X \subseteq V$ with ξ small will allow us to either apply Lemma 3 to halve $\partial(X)$ (possibly after removing S) or halve the size of $|U_X|$ by removing S.

Lemma 6. *Let $\xi \in (0, 1)$, let X be a set with $\emptyset \subsetneq X \subseteq V$, and let S be a ξ-core of X.*

(i) If S is a ξ-witness of X, then X is ξ-balanced.

(ii) If S is a ξ-unbalancer of X, then either $X \setminus S$ contains no 48ξ-unbalancer or $|U_{X \setminus S}| \leq |U_X|/2$.

Proof. Suppose S is a ξ-witness of X, so that either $S = \emptyset$ or $|S| \geq 24/\xi^2$. If $S = \emptyset$, then by (W1) every vertex in X has degree at most $\xi |U_X|$. Since every vertex in U_X is incident to an edge to X, we have $|U_X| \leq \partial(X)$ and so X is ξ-balanced. Suppose instead $|S| \geq 24/\xi^2$. Then by (W2), at least $24/\xi^2$ vertices in X have degree at least $\xi |U_X|/24$. Hence $\partial(X) \geq |U_X|/\xi$. Since every vertex $v \in X$ satisfies $d(v) \leq |U_X|$, it follows that $d(v) \leq \xi \partial(X)$ and so X is ξ-balanced. Thus (i) holds.

Now suppose S is a ξ-unbalancer of X, suppose S' is a 48ξ-unbalancer of $X \setminus S$, and let $v \in S'$ be arbitrary. By (W2), we have $d(v) \geq 2\xi |U_{X \setminus S}|$. Moreover, since $v \notin S$, by (W1) we have $d(v) \leq \xi |U_X|$. It follows that $|U_X| \geq 2|U_{X \setminus S}|$, so (ii) holds. □

We now sketch the proof of the main result. (A full proof is included in [6].)

Algorithm EdgeCount. *Given sets U and V with $|U \cup V| = n$ and a rational $0 < \varepsilon < 1$, return a rational number x such that*

$$(1 - \varepsilon) \cdot e(G) \leq x \leq (1 + \varepsilon) \cdot e(G)$$

holds with probability at least $\frac{2}{3}$.

(E1) Let $\zeta = \varepsilon^2/36^2(\log n)^3$, let $X = V$, and let $t = N = 0$. If $n < 3000$, then count $e(G)$ exactly using adj_G and output it.

(E2) Apply **FindCore** to X and $\zeta/48$. If **FindCore** returns $\partial(X)$, then output $2^t \partial(X) + N$. Otherwise, it returns a subset S of X.

(E3) If $S = \emptyset$ or $|S| > 24 \cdot (48/\zeta)^2$, then remove each element of X independently with probability $1/2$, increment t, and go back to (E2).

(E4) Using adj_G, find $\partial(S)$.

(E5) Apply **FindCore** to $X \setminus S$ with argument ζ. If **FindCore** returns $\partial(X \setminus S)$, then output $2^t \partial(X) + N = 2^t(\partial(S) + \partial(X \setminus S)) + N$. Otherwise, it returns a subset S' of $X \setminus S$.

(E6) If $S' = \emptyset$ or $|S'| > 24/\zeta^2$, then remove S from X, remove each element of X independently with probability $1/2$, add $2^t \partial(S)$ to N, increment t, and go to (E2).

(E7) Otherwise, remove S from X, add $2^t \partial(S)$ to N, and go to (E2).

Theorem 10. *There is a randomised algorithm $\mathcal{A}$ satisfying the following. Suppose that $\mathcal{A}$ is given access to the independence and adjacency oracles of some bipartite graph $G = (U, V, E)$, together with U, V and some rational ε with $0 < \varepsilon < 1$. Then with probability at least $\frac{2}{3}$, algorithm $\mathcal{A}$ returns an ε-approximation to $e(G)$. Moreover, writing $n = |U \cup V|$, algorithm $\mathcal{A}$ runs in time $\varepsilon^{-4} \cdot \tilde{O}(n)$ and calls ind_G at most $\varepsilon^{-2} \cdot O((\log n)^6)$ times.*

Proof sketch. We take $\mathcal{A}$ to be **EdgeCount**, modified to return an arbitrary value if the running time or oracle access exceeds that allowed by Theorem 10.

Suppose without loss of generality that $n \geq 3000$. For all integers $i \geq 1$, let X_i be the value of X at the start of the ith iteration of (E2)–(E7), or $X_i = X_{i-1}$ if the algorithm terminates before this point. Define t_i and N_i in the same fashion. Let S_i be the output of **FindCore** at the ith iteration of (E2), or $\emptyset$ if the algorithm terminates before this point.

We say that the ith iteration of (E2)–(E7) is *good* if one of the following two events occurs:

(i) during the ith iteration, **EdgeCount** halts and outputs

$$2^{t_i} \partial(X_i) + N_i\,,$$

(ii) **EdgeCount** does not halt during the ith iteration, and we have $|X_{i+1}| \cdot |U_{X_{i+1}}| \le (3/4)^{i-1}n^2$ and

$$\left(1 - 2\sqrt{\zeta \log n}\right)^{t_i} e(G) \le 2^{t_i}\partial(X_i) + N_i \le \left(1 + 2\sqrt{\zeta \log n}\right)^{t_i} e(G). \tag{6}$$

Let $\mathcal{E}_i$ be the event that either the ith iteration is good or **EdgeCount** terminates in fewer than i iterations. We will prove that for all $i \ge 1$, we have $\mathbb{P}(\mathcal{E}_i \mid \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{i-1}) \ge 1 - 14/n$.

Suppose $\mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{i-1}$ occurs, and that **EdgeCount** does not terminate in the first $i-1$ iterations (or the claim is trivial). Suppose also that every invocation of **FindCore** in the ith iteration succeeds; by Lemma 5, this holds with probability at least $1-6/n$. In particular, this implies that if **EdgeCount** terminates at (E2) or (E5), then $\mathcal{E}_i$ occurs.

Next, suppose the algorithm loops at (E3). Then S_i is a $(\zeta/48)$-witness for X_i, and so X_i is $(\zeta/48)$-balanced by Lemma 6(i). Moreover, $|X_i| \ge 24 \log n$, or **FindCore** would have returned $\partial(X_i)$ at (E2). It follows by Lemma 3 that with probability at least $1 - 4/n$, we have $|X_{i+1}| \le 3|X_i|/4$ and $|\partial(X_{i+1}) - \partial(X_i)/2| \le \sqrt{\zeta \log n} \cdot \partial(X_i)$. It can easily be shown that this, together with (6) and $N_{i+1} = N_i$, implies that the $(i+1)$st iteration is good. Similarly, if the algorithm loops at (E6), then $X \setminus S_i$ is ζ-balanced and $N_{i+1} = N_i + 2^{t_i}\partial(S_i)$, and so the $(i+1)$st iteration is again good with probability at least $1-4/n$.

Finally, suppose the algorithm loops at (E7). Then S_i is a $(\zeta/48)$-unbalancer of X_i, and the output S_i' of **FindCore** at (E5) is a ζ-unbalancer of $X_i \setminus S_i$. It follows by Lemma 6(ii) that $|U_{X_{i+1}}| = |U_{X_i \setminus S_i}| \le |U_{X_i}|/2$. We have $2^{t_{i+1}}\partial(X_{i+1}) + N_{i+1} = 2^{t_i}\partial(X_i) + N_i$, so it once again follows that the $(i+1)$st iteration is good. Thus by a union bound, we have $\Pr(\mathcal{E}_i \mid \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{i-1}) \ge 1 - 14/n$ as claimed.

Let $m = \lceil 7 \log n \rceil + 1$. By a union bound, since $n \ge 3000$, we have $\Pr(\mathcal{E}_1 \cap \cdots \cap \mathcal{E}_m) \ge 2/3$. Suppose $\mathcal{E}_1 \cap \cdots \cap \mathcal{E}_m$ occurs. This implies that **EdgeCount** terminates within m iterations; indeed, if it has not terminated at the start of the mth, then we must have $|X_m||U_{X_m}| \le (3/4)^{m-1}n^2 < 1$. Thus $X_m = \emptyset$ or $U_{X_m} = \emptyset$; in either case, $\partial(X) = 0$ and so **EdgeCount** will terminate at (E2).

It can easily be verified that since **EdgeCount** terminates within m iterations of (E2)–(E7), the running time and oracle access do not exceed that allowed by Theorem 10, and so **EdgeCount** runs to completion. Moreover, by the definition of ζ we have

$$\left(1 - 2\sqrt{\zeta \log n}\right)^{m} \ge 1 - \varepsilon \quad \text{and} \quad \left(1 + 2\sqrt{\zeta \log n}\right)^{m} \le 1 + \varepsilon,$$

so the output is an ε-approximation of $e(G)$ by (6). □

4 REDUCING APPROXIMATE #k-SAT TO k-SAT

Our result is formally stated as follows. Let $\Pi_{k,s}$ be the satisfiability problem for conjunctions of width-k CNF formulae and width-s XOR clauses, and note that instances of $\Pi_{k,s}$ may be expressed as instances of $\max\{k,s\}$-SAT with the same number of variables. Let $\pi_{k,s}$ be the infimum over all $\beta > 0$ such that $\Pi_{k,s}$ has a randomised algorithm with running time $O^*(2^{\beta n})$ and success probability at least 2/3.

Theorem 11. *Let $k \in \mathbb{N}$ with $k \ge 2$, let $0 < \delta < 1$, and suppose $s \ge 120 \lg(6/\delta)^2/\delta$. Then there is a randomised ε-approximation algorithm for #k-SAT with running time $\varepsilon^{-2} \cdot O\big(2^{(\pi_{k,s}+\delta)n}\big)$.*

We defer a full proof to [6], and instead provide a sketch. For any formula F, let $\mathrm{SAT}(F)$ be the number of satisfying assignments of F. It is relatively easy to show that if F is a k-CNF formula and F' is a collection of m uniformly random XOR clauses on the variables of F, then $\mathrm{SAT}(F \wedge F')$ is concentrated around $2^{-m} \cdot \mathrm{SAT}(F)$ with high probability. Moreover, by a standard self-reducibility argument, the satisfying assignments of $F \wedge F'$ may be counted exactly with $O(\mathrm{SAT}(F \wedge F'))$ invocations of an appropriate decision oracle. Thus by finding an appropriate value of m, one may efficiently obtain a good estimate of $\mathrm{SAT}(F)$. This is the classical argument of Valiant and Vazirani [25].

However, this proof requires modification to work in the exponential setting. If F has n variables, then each uniformly random XOR clause has width $\Theta(n)$ with high probability, and therefore cannot be expressed as a width-k CNF without introducing $\Omega(n)$ new variables. Thus $F \wedge F'$ may 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 approximate counting algorithm for #k-SAT. In fact, we can afford to add only constant-width XORs, which do not in general result in concentration in $\mathrm{SAT}(F \wedge F')$.

We therefore make use of a hashing scheme developed by Calabro, Impagliazzo, Kabanets, and Paturi [3] for a related problem, that of reducing k-SAT to Unique-k-SAT. In our terminology, an (s,m,n)-*hash* is a random $m \times n$ matrix A over 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 $A_{i,j} \in \mathrm{GF}(2)$ independently and uniformly at random, and set all other entries of A to zero. Taking $\boldsymbol{x}$ to be the length-n vector of variables appearing in F, we then choose $\boldsymbol{b} \in \mathrm{GF}(2)^m$ uniformly at random and take F' to be $(A\boldsymbol{x} = \boldsymbol{b})$. This still does not yield concentration in $\mathrm{SAT}(F \wedge F')$ when $s = O(1)$, but it turns out the variance is sufficiently low that we can obtain concentration by summing over many slightly stronger independently-chosen hashes. The key lemma (which we prove in [6]) is the following.

Lemma 7. *Let $0 < \delta < 1/6$ and let $s, m, n \in \mathbb{N}$. Suppose $m \le n$ and $s \ge 20 \lg(1/\delta)^2/\delta$. Let $S \subseteq \mathrm{GF}(2)^n$ and suppose $|S| \ge 2^{m+\delta n}$. Let A be an (s,m,n)-hash, and let $\boldsymbol{b} \in \mathrm{GF}(2)^m$ be uniformly random and independent of A. Let $S' = \{\boldsymbol{x} \in S : A\boldsymbol{x} = \boldsymbol{b}\}$. Then $\mathbb{E}(|S'|) = 2^{-m}|S|$ and $\mathrm{Var}(|S'|) \le |S|^2 2^{\delta n/16 - 2m}$.*

The remainder of the proof proceeds as in [25].

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 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*. 218–230.

[2] Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. 2008. Subquadratic Algorithms for 3SUM. *Algorithmica* 50, 4 (01 Apr 2008), 584–596. https://doi.org/10.1007/s00453-007-9036-3

[3] 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.

[4] 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*. 31–40.

[5] Timothy M. Chan and 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*. 1246–1255.

[6] Holger Dell and John Lapinskas. 2018. Fine-grained reductions from approximate counting to decision. *CoRR* (2018). abs/1707.04609.

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

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

[9] Anka Gajentaan and Mark H Overmars. 1995. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry* 5, 3 (1995), 165–185.

[10] Timon Hertli. 2014. 3-SAT Faster and Simpler - Unique-SAT Bounds for PPSZ Hold in General. *SIAM J. Comput.* 43, 2 (2014), 718–729.

[11] 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*. 961–972.

[12] Russell Impagliazzo and Ramamohan Paturi. 2001. On the Complexity of k-SAT. *J. Comput. Syst. Sci.* 62, 2 (2001), 367–375.

[13] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.* 63, 4 (2001), 512–530.

[14] Konstantin Kutzkov. 2007. New upper bound for the #3-SAT problem. *Inf. Process. Lett.* 105, 1 (2007), 1–5.

[15] Colin McDiarmid. 1989. On the method of bounded differences. In *Surveys in Combinatorics, 1989: Invited Papers at the Twelfth British Combinatorial Conference*. 148–188.

[16] 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*. 50–59.

[17] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. 2005. An Improved Exponential-time Algorithm for k-SAT. *J. ACM* 52, 3 (May 2005), 337–364.

[18] Mihai Pătraşcu. 2010. Towards Polynomial Lower Bounds for Dynamic Problems. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing (STOC '10)*. ACM, New York, NY, USA, 603–610.

[19] Michael Sipser. 1983. A Complexity Theoretic Approach to Randomness. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing (STOC '83)*. New York, NY, USA, 330–335.

[20] Larry Stockmeyer. 1985. On Approximation Algorithms for #P. *SIAM J. Comput.* 14, 4 (1985), 849–861.

[21] 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. 78–87.

[22] Seinosuke Toda. 1991. PP is As Hard As the Polynomial-time Hierarchy. *SIAM J. Comput.* 20, 5 (Oct. 1991), 865–877.

[23] Patrick Traxler. 2014. The Relative Exponential Time Complexity of Approximate Counting Satisfying Assignments. In *Parameterized and Exact Computation: 9th International Symposium, IPEC 2014, Wroclaw, Poland, September 10-12, 2014. Revised Selected Papers*. 332–341.

[24] Leslie G. Valiant. 1979. The complexity of computing the permanent. *Theor. Comput. Sci.* 8, 2 (1979), 189–201.

[25] Leslie G. Valiant and Vijay V. Vazirani. 1986. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.* 47 (1986), 85–93.

[26] Ryan Williams. 2005. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.* 348, 2-3 (2005), 357–365.

[27] 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*. 664–673.

[28] 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*. 1867–1877.

[29] Virginia V. Williams. 2015. Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis. In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*. 17–29.

[30] Virginia V. Williams and Ryan Williams. 2010. Subcubic Equivalences between Path, Matrix and Triangle Problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. 645–654.