On Super Strong ETH

Nikhil Vyas(✉) and Ryan Williams

MIT, Cambridge, MA, USA
{nikhilv,rrw}@mit.edu

Abstract. Multiple known algorithmic paradigms (backtracking, local search and the polynomial method) only yield a $2^{n(1-1/O(k))}$ time algorithm for k-SAT in the worst case. For this reason, it has been hypothesized that the worst-case k-SAT problem cannot be solved in $2^{n(1-f(k)/k)}$ time for any unbounded function f. This hypothesis has been called the "Super-Strong ETH", modeled after the ETH and the Strong ETH. We give two results on the Super-Strong ETH:

1. It has also been hypothesized that k-SAT is hard to solve for randomly chosen instances near the "critical threshold", where the clause-to-variable ratio is $2^k \ln 2 - \Theta(1)$. We give a randomized algorithm which refutes the Super-Strong ETH for the case of random k-SAT and planted k-SAT for any clause-to-variable ratio. For example, given any random k-SAT instance F with n variables and m clauses, our algorithm decides satisfiability for F in $2^{n(1-\Omega(\log k)/k)}$ time, with high probability (over the choice of the formula and the randomness of the algorithm). It turns out that a well-known algorithm from the literature on SAT algorithms does the job: the PPZ algorithm of Paturi, Pudlák and Zane [17].
2. The Unique k-SAT problem is the special case where there is at most one satisfying assignment. Improving prior reductions, we show that the Super-Strong ETHs for Unique k-SAT and k-SAT are equivalent. More precisely, we show the time complexities of Unique k-SAT and k-SAT are very tightly correlated: if Unique k-SAT is in $2^{n(1-f(k)/k)}$ time for an unbounded f, then k-SAT is in $2^{n(1-f(k)(1-\varepsilon)/k)}$ time for every $\varepsilon > 0$.

1 Introduction

The k-SAT problem is the canonical NP-complete problem for $k \geq 3$. Tremendous effort has been devoted to finding faster worst-case algorithms for k-SAT. Because it is widely believed that $\mathsf{P} \neq \mathsf{NP}$, the search has been confined to super-polynomial-time algorithms. Despite much effort, there are no known algorithms for k-SAT which run in $(2-\epsilon)^n$ time for a universal constant $\epsilon > 0$, independent of k. This inability to find algorithms has led researchers to the following two popular hypotheses, which strengthen $\mathsf{P} \neq \mathsf{NP}$:

N. Vyas—Supported by NSF Grant CCF-1741615.
R. Williams—Supported by NSF CCF-1741615.

M. Janota and I. Lynce (Eds.): SAT 2019, LNCS 11628, pp. 406–423, 2019.
https://doi.org/10.1007/978-3-030-24258-9_28

- **Exponential Time Hypothesis (ETH)** [14] There is an $\alpha > 0$ such that no 3-SAT algorithm runs in $2^{\alpha n}$ time.
- **Strong Exponential Time Hypothesis (SETH)** [5] There does not exist a constant $\epsilon > 0$ such that for all k, k-SAT can be solved in $(2-\epsilon)^n$ time.

The present situation for worst-case k-SAT algorithms looks even worse than hypothesized. The current best known algorithms for k-SAT all have running time bounds of the form $2^{n\left(1-\Omega\left(\frac{1}{k}\right)\right)}$, i.e., $2^{n\left(1-\frac{c}{k}\right)}$ for some constant $c > 0$. It is a very interesting phenomenon that the same running time upper bound is achieved by radically different algorithmic paradigms, such as randomized backtracking [16,17], local search [19], the polynomial method [6] and linear programming based methods [3]. Even for simpler variants such as unique-k-SAT, no significantly faster algorithms are known (with a better dependence on k in the exponent). Hence it is possible that the runtime behavior of $2^{n\left(1-\Omega\left(\frac{1}{k}\right)\right)}$ is actually optimal for k-SAT algorithms. This was termed the Super-Strong ETH in a 2015 talk by the second author [24]. We state the Super-SETH as follows:

Super-SETH: Super Strong Exponential Time Hypothesis
For every unbounded function $f : \mathbb{N} \to \mathbb{N}$, there is no (randomized) algorithm for k-SAT running in $2^{n\left(1-\frac{f(k)}{k}\right)}$ time.

Intuitively, Super-SETH says that the $\Omega(1/k)$ "savings" in the exponent is optimal: not even an $f(k)/k$ savings can be achieved, for any unbounded f. In this paper, we study Super-SETH in two natural restricted scenarios:

- **Random/Planted k-SAT**. We consider two general cases: (a) finding solutions to random k-SAT instances where each clause is drawn uniformly and independently from all possible k-width clauses, and (b) finding solutions to planted k-SAT instances, where a random (hidden) solution σ is sampled, then each clause is drawn uniformly and independently from all possible k-width clauses that satisfy σ.
  Random k-SAT has a well-known threshold behaviour in which, for $\alpha_{sat} = 2^k \ln 2 - \Theta(1)$ and for all constant $\epsilon > 0$, random k-SAT instances are SAT w.h.p. (with high probability) for $m < (\alpha_{sat} - \epsilon)n$ and UNSAT w.h.p. for $m > (\alpha_{sat} + \epsilon)n$. Note that, as far as decidability is concerned, for instances below (respectively, above) the threshold we may simply output "SAT" (respectively, "UNSAT") and we will be correct whp. It has been conjectured [10,20] that random instances at the threshold $m = \alpha_{sat} n$ are the hardest random instances, and it is difficult to determine their satisfiability. We are motivated by the following strengthening of this conjecture: **Are random instances near the threshold as hard as the worst-case instances of k-SAT?**
- **Unique k-SAT**. This is the special case of finding a SAT assignment to a k-CNF, when one is promised that there is at most one satisfying assignment. It is well-known to be NP-complete under randomized reductions [22]. As

mentioned earlier, the best known algorithms for Unique-k-SAT have the same running time behaviour of $2^{n\left(1-O\left(\frac{1}{k}\right)\right)}$ as k-SAT. In fact some of the best-known k-SAT algorithms [16,17] have an easier analysis when restricted to the case of Unique-k-SAT. PPSZ [16], the current best known algorithm for k-SAT (when $k \geq 5$) has only been derandomized for Unique-k-SAT. **Could worst-case algorithms for Unique k-SAT be marginally faster than those for k-SAT?**

In principle, in this "ultra fine-grained" setting we are studying (where the exponential dependence on k matters), both above special cases could potentially be just as hard as k-SAT, or both of them could be easier. In this paper, we prove that Super-SETH is false for Random k-SAT, and the Super-SETH for Unique k-SAT is equivalent to the general Super-SETH: the dependence on k in the exponent is the *same* for the two problems.

1.1 Prior Work

As mentioned earlier, many algorithmic paradigms have been introduced for solving k-SAT in the worst case, but none are known to run in $2^{n(1-\omega_k(1/k))}$ time. There also has been substantial work on polynomial-time algorithms for random k-SAT that return solutions for m below the threshold. Note that even though we know that these instances are satisfiable whp, that does not immediately give a way to *find* a solution. Chao and Franco [7] first proved that the unit clause heuristic (the same key component of the PPZ algorithm) finds solutions with high probability for random k-SAT when $m \leq c2^k n/k$ for some constant $c > 0$. The current best known polynomial-time algorithm in this regime is by Coja-Oghlan [8] and it can find a solution whp for random k-SAT when $m \leq c2^k n \log(k)/k$ for some constant $c > 0$. Interestingly, we also know of polynomial time algorithms for large m. Specifically, it is known that for a certain constant $C_0 = C(k)$ and $m > C_0 \cdot n$ there are polynomial-time algorithms finding solutions to planted k-SAT instances by Krivelevich and Vilenchik [15] and random k-SAT (conditioned on satisfiability) by Coja-Oghlan, Krivelevich and Vilenchik [9]. However, both of these results require that $m \geq 4^k n/k$ [23]. To our knowledge, no improvements over worst-case k-SAT algorithms have yet been reported for random k-SAT very close to the threshold.

Valiant and Vazirani [22] gave poly-time randomized reductions from SAT instances F on n variables to Unique-SAT instances F' on n variables such that, if F is SAT then F' a unique satisfying assignment with probability at least $\Omega(1/n)$, and if F is UNSAT then F' is UNSAT. This reduction is not applicable to convert k-SAT instances to Unique-k-SAT instances, as they do not preserve the clause width (and when we perform a reduction to reduce the clause width, the number of variables blows up too much for exponential-time algorithms). To address this, Calabro, Impagliazzo, Kabanets and Paturi [4] gave a randomized polynomial-time reduction with one-sided error from k-SAT to Unique-k-SAT which works with probability $2^{-O(n \log^2(k)/k)}$. The probability bound was further

improved by Traxler [21] to $2^{-O(n \log(k)/k)}$. Both of these reductions imply that k-SAT and Unique k-SAT either both have $2^{\delta n}$ time algorithms for some universal $\delta < 1$, or neither of them do (i.e., the SETH and the SETH for Unique-k-SAT are equivalent). However these results are not sufficient for an equivalence w.r.t. Super-SETH: for example, from these results it is still possible that k-SAT has no $2^{n(1-\omega_k(1/k))}$ time algorithms, while Unique-k-SAT has a $2^{n(1-\Omega(\log k/k))}$ time algorithm.

1.2 Our Results

Average-Case k-SAT Algorithms. First we present an algorithm which breaks the Super-Strong ETH for random k-SAT. In particular, we give a $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$-time algorithm which finds a solution whp for random-k-SAT (conditioned on satisfiability) for all values of m. In fact, our algorithm is an old one from the SAT algorithms literature: the PPZ algorithm of Paturi, Pudlak and Zane [17].

In order to show that PPZ breaks Super-Strong ETH in the random case, we first show that PPZ yields a faster algorithm for random *planted* k-SAT for large enough m.

Theorem 1. *There is a randomized algorithm that, given a* **planted** *k-SAT instance F sampled from $P(n,k,m)$[1] with $m > 2^{k-1}\ln(2)$, outputs a satisfying assignment to F in $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$ time with $1-2^{-\Omega\left(n\left(\frac{\log k}{k}\right)\right)}$ probability (over the planted k-SAT distribution and the randomness of the algorithm).*

Next, we give a reduction from random k-SAT (conditioned on satisfiability, we denote this distribution by R^+) to planted k-SAT. Similar reductions/equivalences have been observed before in [1,2].

Theorem 2. *Suppose there is an algorithm A for planted k-SAT $P(n,k,m)$, for all $m \geq 2^k \ln 2(1-f(k)/2)n$, which finds a solution in time $2^{n(1-f(k))}$ and with probability $1-2^{-nf(k)}$, where $1/k < f(k) = o_k(1)$. Then for any m', given a random k-SAT instance sampled from $R^+(n,k,m')$, a satisfying assignment can be found in $2^{n(1-\Omega(f(k)))}$ time whp.*

Combining Theorems 1 and 2 yields:

Theorem 3. *Given a random k-SAT instance F sampled from $R^+(n,k,m)$, we can find a solution in $2^{n(1-\Omega(\frac{\log k}{k}))}$ time whp.*

Remark 1. We obtain a randomized algorithm for random k-SAT which always reports UNSAT on unsatisfiable instances, and finds a SAT assignment whp on satisfiable instances. Feige's Hypothesis for k-SAT [12] conjectures that there are no efficient *refutations* for random k-SAT near the threshold, i.e., there are no

[1] See "Three k-SAT Distributions" in Sect. 2 for formal definitions of different k-SAT distributions.

efficient algorithms which always report SAT on satisfiable instances, and report UNSAT on unsatisfiable instances with probability at least 1/2. Refuting Feige's hypothesis in our setting remains an intriguing open problem.

Theorems 1 and 3 imply that at least one of the following are true:

- Either the random instances of k-SAT at the threshold are *not* the hardest instances of k-SAT, or
- Super-Strong ETH is false.

For the PPZ algorithm (randomized branching with unit propagation), time **lower bounds** of the form $2^{n(1-O(\frac{1}{k}))}$ are in fact known [18]. Thus we can say that, with respect to the PPZ algorithm, random k-SAT instances are *provably* more tractable than worst-case k-SAT instances. On the other hand, for the PPSZ algorithm (randomized branching with limited resolution on small sets of clauses) which gives the current best known running time for k-SAT (when $k \geq 4$) we only know $2^{n\left(1-O\left(\frac{\log k}{k}\right)\right)}$ lower bounds [18], matching our upper bounds for the random case. Hence it is possible that PPSZ actually runs in $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$ time for worst-case k-SAT.

We observe that our techniques can be used to get algorithms running faster than $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$ for planted k-SAT and random k-SAT (conditioned on satisfiability), depending on how large m/n is compared to the threshold density. These results appear in the full version.

Unique k-SAT Equivalence. In Sect. 5 we give a "low exponential" time reduction from k-SAT to Unique-k-SAT, which proves that the two problems are equivalent w.r.t. Strong-SETH: i.e., there is a $2^{n(1-\omega_k(1/k))}$ time algorithm for Unique-k-SAT if and only if there is a $2^{n(1-\omega_k(1/k))}$ time algorithm for k-SAT. In fact, our reduction has the following stronger property:

Theorem 4. *If Unique k-SAT is solvable in $2^{(1-f(k)/k)n}$ time for some unbounded function f, then k-SAT is solvable in $2^{(1-f(k)/k+O(\log(f(k))/k))n}$ time.*

As mentioned earlier, the current best algorithm for k-SAT PPSZ [16] has a much easier analysis for Unique k-SAT, and in fact it was an open question to show that its running time on general instances of k-SAT matches the running time for Unique k-SAT; this was eventually resolved by Hertli [13]. Theorem 4 implies that, in order to obtain faster algorithms for k-SAT which break Super-Strong ETH, it would be sufficient to restrict ourselves to Unique k-SAT, which might simplify the analysis as in the case of PPSZ.

2 Preliminaries

Notation. In this paper, we generally assume $k \geq 3$ is an arbitrarily large integer. Throughout the paper, we compare time bounds that have the form

$2^{n(1-\Omega(\log k)/k)}$ with $2^{n(1-O(1/k))}$ time, where the big-Ω and the big-O hide multiplicative constants; such notation only makes sense when k can grow unboundedly.

We use the terms "solution", "SAT assignment", and "satisfying assignment" interchangeably. For an n-variable assignment $s \in \{0,1\}^n$ and an index set $I \subseteq [n]$, we use $s_{|I}$ to denote the length-$|I|$ substring of s projected on the index set I. We use the notation $x \in_r \chi$ to denote that x is randomly sampled from the distribution χ. By $poly(n)$, we mean some function $f(n)$ which satisfies $f(n) = O(n^c)$ for a universal constant $c \geq 1$. Letting n be the number of variables in a k-CNF, a random event about k-CNF holds *whp* (with high probability) if it holds with probability $1 - f(n)$, where $f(n) \to 0$ as $n \to \infty$. We use log and ln to denote the logarithm base-2 and base-e respectively, and $H(p) = -p\log(p) - (1-p)\log(1-p)$ denotes the binary entropy function, and $\tilde{O}(f(n))$ denotes $O(f(n)\log(f(n)))$.

Three k-SAT Distributions. We consider three distributions for random k-SAT:

- $R(n,k,m)$ is the distribution over formulas F of m clauses, where each clause is drawn i.i.d. from the set of all k-width clauses. This is the standard k-SAT distribution.
- $R^+(n,k,m)$ is the distribution over formulas F of m clauses where each clause is drawn i.i.d. from the set of all k-width clauses and we condition F on being satisfiable i.e. $R(n,k,m)$ conditioned on satisfiability.
- $P(n,k,m,\sigma)$ is the distribution over formulas F of m clauses where each clause is drawn i.i.d. from the set of all k-width clauses which satisfy σ. $P(n,k,m)$ is the distribution over formulas F formed by sampling $\sigma \in \{0,1\}^n$ uniformly and then sampling F from $P(n,k,m,\sigma)$.

Note that an algorithm solving the search problem (finding SAT assignments) for instances sampled from R^+ is stronger than deciding satisfiability for instances sampled from R: given an algorithm for the search problem on R^+, we may run it on a random instance from R, returning SAT if and only if the algorithm returns a valid satisfying assignment.

2.1 Structural Properties of Planted and Random k-SAT

A few structural results about planted and random k-SAT will be useful in analyzing our algorithms. In particular, we consider bounds on the expected number of solutions of planted k-SAT instances and random k-SAT instances (conditioned on satisfiability).

A well-known conjecture is that the satisfiability of random k-SAT displays a threshold behaviour for all k. The following lemma which states that the conjecture holds for all k (larger than a fixed constant) was proven by Ding, Sly and Sun [11].

Lemma 1 ([11]). *There is a constant k_0 such that for all $k > k_0$, there exists an $\alpha_{sat} = 2^k \ln 2 - \Theta(1)$ and for all constant $\epsilon > 0$, we have that:*

$$\text{For } m < (1-\epsilon)\alpha_{sat} n, \lim_{n\to\infty} \Pr_{F \in_r R(n,k,m)}[F \text{ is satisfiable}] = 1$$

$$\text{For } m > (1+\epsilon)\alpha_{sat} n, \lim_{n\to\infty} \Pr_{F \in_r R(n,k,m)}[F \text{ is satisfiable}] = 0$$

We will also need the fact that, whp, the ratio of the number of solutions and its expected value is not too small, as long as m is not too large.

Lemma 2 (Achlioptas, Lemma 22 of [1]). *For $F \in_r R(n,k,m)$, let $\mathcal{S}$ be the set of solutions of F. Then $E[|\mathcal{S}|] = 2^n(1-\frac{1}{2^k})^m$. Furthermore, for $\alpha_d = 2^k \ln 2 - k$ and $m < \alpha_d n$ we have*

$$\lim_{n\to\infty} \Pr[|\mathcal{S}| < E[|\mathcal{S}|]/2^{O(nk/2^k)}] = 0.$$

Together, the above two results have the following useful consequence. Intuitively, the below lemma states that if our random k-SAT instance is slightly below the threshold, then (conditioned on being satisfiable) we can fairly tightly bound the expected number of SAT assignments.

Lemma 3. *For $F \in_r R^+(n,k,m)$ let Z denote the number of solutions of F. For all constant $\delta > 0$, if $m < (1-\epsilon)\alpha_{sat}$ for some constant $\epsilon > 0$, then $2^n(1-\frac{1}{2^k})^m \leq E[Z] \leq (1+\delta)2^n(1-\frac{1}{2^k})^m$. Furthermore, for $\alpha_d = 2^k \ln 2 - k$ and $m < \alpha_d n$,*

$$\lim_{n\to\infty} \Pr[Z < E[Z]/2^{O(nk/2^k)}] = 0.$$

Proof. Let $F' \in_r R(n,k,m)$ and let Z' denote the number of solutions of F'. Letting p_n denote the probability that F' is unsatisfiable, we have $E[Z'] = (1-p_n)E[Z]$. By Lemma 1, $\lim_{n\to\infty} p_n \to 0$, hence $2^n(1-\frac{1}{2^k})^m \leq E[Z] \leq (1+\delta)2^n(1-\frac{1}{2^k})^m$.

Observe that $\Pr[Z < E[Z]/2^{O(nk/2^k)}] \leq \Pr[Z' < E[Z]/2^{O(nk/2^k)}]$, as Z is just Z' conditioned on being positive. Furthermore $\Pr[Z' < E[Z]/2^{O(nk/2^k)}] \leq \Pr[Z' < E[Z']/2^{O(nk/2^k)}]$ as $E[Z] \leq 2E[Z']$. By Lemma 2, $\Pr[Z' < E[Z']/2^{O(nk/2^k)}]$ tends to 0. □

We will use a planted k-SAT algorithm to solve random k-SAT instances conditioned on their satisfiability. The basic idea was introduced in an unpublished manuscript by Ben-Sasson, Bilu, and Gutfreund [2]. We will use the following lemma therein.

Lemma 4 (Lemma 3.3 of [2]). *For a given F in $R^+(n,k,m)$ with Z solutions, it is sampled from $P(n,k,m)$ with probability Zp, where p only depends on n, k, and m.*

Corollary 1. *For $F \in_r R^+(n,k,m)$ and $F' \in_r P(n,k,m)$ let Z and Z' denote their number of solutions respectively. Then for $\alpha_d = 2^k \ln 2 - k$ and for $m < \alpha_d n$, $\lim_{n\to\infty} \Pr[Z' < E[Z]/2^{O(nk/2^k)}] = 0$.*

Proof. We have $\lim_{n\to\infty} \Pr[Z < E[Z]/2^{O(nk/2^k)}] = 0$ by Lemma 3. Lemma 4 shows that the planted k-SAT distribution $P(n,k,m)$ is biased toward satisfiable formulas with more solutions. The distribution $R^+(n,k,m)$ instead chooses all satisfiable formulas with equal probability. Hence $\lim_{n\to\infty} \Pr[Z' < E[Z]/2^{O(nk/2^k)}] = 0$. □

So far, our lemmas have only handled the case where $m > \alpha_{sat} n$. Next we prove a lemma bounding the number of expected solutions when $m > \alpha_{sat} n$.

Lemma 5. *For $m \geq (\alpha_{sat} - 1)n$, the expected number of solutions of $F \in_r R^+(n,k,m)$ and $F' \in_r P(n,k,m)$ is at most $2^{O(n/2^k)}$ in both cases.*

Proof. Lemma 4 shows that the planted k-SAT distribution $P(n,k,m)$ is biased toward satisfiable formulas with more solutions. In particular, the expected number of solutions of $F' \in_r P(n,k,m)$ upper bounds the expected number for $F \in_r R^+(n,k,m)$. So it suffices to upper bound the expected number of solutions of $F' \in_r P(n,k,m)$.

Let Z be the random variable denoting the number of solutions of F'. Let σ denote the planted solution in F, and let x be some assignment which has hamming distance i from σ. For a clause C satisfied by σ but not by x, all of C's satisfied literals must come from the i bits where σ and x differ, and all its unsatisfied literals must come from the remaining $n - i$ bits. Letting j denote the number of satisfying literals in C, the probability that a randomly sampled clause C is satisfied by σ but not by x is $\sum_{j=1}^{k} \frac{\binom{k}{j}}{2^k-1} (\frac{i}{n})^j (1 - \frac{i}{n})^{k-j} = \frac{1-(1-\frac{i}{n})^k}{2^k-1}$. We will now upper bound $E[Z]$.

$$
\begin{aligned}
E[Z] &= \sum_{y \in \{0,1\}^n} \Pr[y \text{ satisfies } F'] \\
&= \sum_{i=1}^{n} \binom{n}{i} \Pr[\text{Assignment } x \text{ that differs from } \sigma \text{ in } i \text{ bits satisfies } F'] \\
&= \sum_{i=1}^{n} \binom{n}{i} \Pr[\text{A random clause satisfying } \sigma \text{ satisfies } x]^m \\
&= \sum_{i=1}^{n} \binom{n}{i} (1 - \Pr[\text{A random clause satisfying } \sigma \text{ does not satisfy } x])^m \\
&= \sum_{i=1}^{n} \binom{n}{i} \left(1 - \frac{1-(1-i/n)^k}{2^k-1}\right)^m \text{ [As shown above]} \\
&\leq \sum_{i=1}^{n} \binom{n}{i} e^{-m\left(\frac{1-(1-i/n)^k}{2^k-1}\right)} \text{ [As } 1 - x \leq e^{-x}\text{]} \\
&\leq \sum_{i=1}^{n} \binom{n}{i} e^{-(\alpha_{sat}-1)n\left(\frac{1-(1-i/n)^k}{2^k-1}\right)}
\end{aligned}
$$

$$\leq 2^{O(n/2^k)} \sum_{i=1}^{n} \binom{n}{i} e^{-((2^k-1)\ln 2)n\left(\frac{1-(1-i/n)^k}{2^k-1}\right)} \text{ [As } m \geq (2^k \ln 2 - O(1))n]$$

$$\leq 2^{O(n/2^k)} \sum_{i=1}^{n} \binom{n}{i} 2^{-n\left(1-(1-i/n)^k\right)}$$

$$\leq 2^{O(n/2^k)} \sum_{i=1}^{n} 2^{n\left(H(i/n)-1+(1-i/n)^k\right)} \leq 2^{O(n/2^k)} \max_{0\leq p\leq 1} 2^{n\left(H(p)-1+(1-p)^k\right)}.$$

Let $f(p) = H(p) - 1 + (1-p)^k$. Then $f'(p) = -\log\left(\frac{p}{1-p}\right) - k(1-p)^{k-1}$ and $f''(p) = \frac{-1}{p(1-p)} + k(k-1)(1-p)^{k-2}$. Thus $f''(p) = 0 \iff p(1-p)^{k-1} = \frac{1}{k(k-1)}$. Note that $f''(p)$ has only two roots in $[0,1]$, hence $f'(p)$ has at most 3 roots in $[0,1]$. It can be verified that for sufficiently large k, $f'(p)$ indeed has three roots at $p = \Theta(1/2^k)$, $\Theta(\log k/k)$, and $1/2 - \Theta(k/2^k)$. At all these three values of p, $f(p) = O(1/2^k)$. Hence $E[Z] \leq 2^{O(n/2^k)}$. □

3 Planted k-SAT and the PPZ Algorithm

In this section, we establish that the PPZ algorithm solves random planted k-SAT instances faster than $2^{n-n/O(k)}$ time.

Reminder of Theorem 1. *There is a randomized algorithm that given a planted k-SAT instance F sampled from $P(n,k,m)$ with $m > 2^{k-1}\ln(2)$, outputs a satisfying assignment to F in $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$ time with $1 - 2^{-\Omega\left(n\left(\frac{\log k}{k}\right)\right)}$ probability (over the planted k-SAT distribution and the randomness of the algorithm).*

We will actually prove a stronger claim:

> For any σ and F sampled from $P(n,k,m,\sigma)$, we can find a set S of $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$ variable assignments in $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$ time, such that $\sigma \in S$ with probability $1 - 2^{-\Omega\left(n\left(\frac{\log k}{k}\right)\right)}$ (the probability is over the planted k-SAT distribution and the randomness of the algorithm).

Theorem 1 yields an algorithm that (always) finds a solution for k-SAT instance F sampled from $P(n,k,m)$, and runs in *expected* time $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$. In fact, the algorithm of Theorem 1 is a simplification of the PPZ algorithm [17], a well-known worst case algorithm for k-SAT. PPZ runs in polynomial time, and outputs a SAT assignment (on any satisfiable k-CNF) with probability $p \geq 2^{-n+n/O(k)}$. It can be repeatedly run for $O(n/p)$ times to obtain a worst-case algorithm that is correct whp. We consider a simplified version which is sufficient for analyzing planted k-SAT:

Algorithm 1. Algorithm for planted k-SAT

```
procedure SIMPLE-PPZ(F)
    while i ≤ n do
        if there is a unit clause C in the formula then
            Assign the variable in C so that C is true
        else if x_i is unassigned then
            Assign x_i randomly. Set i ← i + 1
        else
            Set i ← i + 1
    Output the assignment if it satisfies F.
```

Our Simple-PPZ algorithm (Algorithm 1) only differs from PPZ in that PPZ also performs an initial random permutation of variables. For us, a random permutation is unnecessary: a random permutation of the variables in the planted k-SAT distribution yields the same distribution of instances. That is, the original PPZ algorithm would have the same behavior as Simple-PPZ on random instances.

We will start with a few useful definitions.

Definition 1 ([17])**.** *A clause C is* critical *with respect to variable x_i and SAT assignment σ if x_i is the only variable in C whose corresponding literal is satisfied by σ.*

Definition 2. *A variable x_i in F is* good *for an assignment σ if there is a clause C in F which is critical with respect to x_i and σ, and i is the largest index among all variables in C. We say that x_i is* good with respect to C *in such a case. A variable which is not good is called* bad*.*

Observe that for every good variable x_i, if all variables x_j for $j < i$ are assigned correctly with respect to σ, then Simple-PPZ sets x_i correctly, due to the unit clause rule. As such, given a formula F with z good variables for σ, the probability that Simple-PPZ finds σ is at least $2^{-(n-z)}$: if all $n-z$ bad variables are correctly assigned, the algorithm is forced to set all good variables correctly as well. Next, we prove a high-probability lower bound on the number of good variables in a random planted k-SAT instance.

Lemma 6. *For $m > n2^{k-1}\ln 2$, a planted k-SAT instance sampled from $P(n,k,m,\sigma)$ has $\Omega(n\log k/k)$ good variables with respect to σ, with probability $1-2^{-\Omega\left(\frac{n\log k}{k}\right)}$.*

Proof. Let $F \in_r P(n,k,m,\sigma)$ and let L be the last (when sorted by index) $n\ln k/(2k)$ variables. Let L_g, L_b be the good and bad variables respectively, with respect to σ, among the variables in L. Let E be the event that $|L_g| \leq n\ln k/(500k)$. Our goal is to prove a strong upper bound on the probability that E occurs. For all $x_i \in L$, we have that $i \geq n(1-\ln k/(2k))$. Observe that if a clause C is such that $x_i \in L_b$ is good with respect to C, then C does not occur

in F. We will lower bound the probability of such a clause occurring in F, with respect to a fixed variable $x_i \in L$. Recall that in planted k-SAT, each clause is drawn uniformly at random from the set of clauses satisfied by σ. Fixing σ and a variable x_i and sampling one clause C, we get that

$$\begin{aligned}
&\Pr_{C \text{ which satisfies } \sigma}[x_i \in L \text{ is good with respect to } C] \\
&= \frac{\text{number of clauses for which } x_i \in L \text{ is good}}{\text{total number of clauses satisfying } \sigma} = \frac{\binom{i-1}{k-1}}{\binom{n}{k}(2^k-1)} \\
&\geq \frac{1}{2}\left(\frac{i}{n}\right)^{k-1} \frac{k}{2^k n} [\text{As } i \geq n(1-\ln k/(2k))] \\
&\geq \frac{1}{2}\left(\frac{i}{n}\right)^{k} \frac{k}{2^k n} \geq \frac{1}{2}\left(1-\frac{\ln k}{2k}\right)^{k} \frac{k}{2^k n} [\text{As } i \geq n(1-\ln k/(2k))] \\
&\geq \frac{1}{2}\left(e^{-\ln k/k}\right)^{k} \frac{k}{2^k n} \text{ [As } k \text{ is large, and } e^{-w} \leq 1-w/2 \text{ for small enough } w>0] \\
&\geq \frac{1}{2^{k+1} n}
\end{aligned}$$

If the event E is true, then $|L_b| > n \ln k/(4k)$. Therefore, *every time we sample a clause* C, the probability that C makes some variable $x_i \in L_b$ good is at least $\frac{\ln k}{k2^{k+3}}$, as the sets of clauses which make different variables good are disjoint sets. Now we upper bound the probability of E occurring:

$$\begin{aligned}
\Pr[E] &\leq \sum_{i=1}^{n \ln k/(500k)} \Pr[\text{exactly } i \text{ vars among the last } n \ln k/(2k) \text{ vars are good}] \\
&\leq \sum_{i=1}^{n \ln k/(500k)} \binom{n \ln k/(2k)}{i} \left(1-\frac{\ln k}{k2^{k+3}}\right)^{m} \\
&\leq n\binom{n \ln k/(2k)}{n \ln k/(500k)} \left(1-\frac{\ln k}{k2^{k+3}}\right)^{n2^{k-1}\ln 2} .[\text{As } m > n2^{k-1}\ln 2]
\end{aligned}$$

Applying the inequality $1-x \leq e^{-x}$ for $x > 0$, the above is at most

$$n\binom{n \ln k/(2k)}{n \ln k/(500k)} \left(e^{-\frac{\ln k}{k2^{k+3}}}\right)^{n2^{k-1}\ln 2} \leq n\binom{n \ln k/(2k)}{n \ln k/(500k)} \left(2^{-\frac{n \ln k}{16k}}\right) \leq 2^{-\delta \frac{n \ln k}{k}}$$

for appropriately small but constant $\delta > 0$, which proves the lemma statement. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. By Lemma 6, we know that with probability at least $1-p$ for $p = 2^{-\Omega\left(n\left(\frac{\log k}{k}\right)\right)}$, the number of good variables with respect to a hidden planted solution σ in F is at least $\gamma n \log k/k$ for a constant $\gamma > 0$. For such instances, a single run of PPZ outputs σ with probability at least $2^{-n(1-\gamma \log k/k)}$. Repeating

PPZ for poly$(n)2^{n(1-\gamma \log k/k)}$ times implies a success probability at least $1-1/2^n$. Hence the overall error probability is at most $p + 1/2^n \leq 2^{-\Omega\left(n\left(\frac{\log k}{k}\right)\right)}$. □

We proved that PPZ runs in time $2^{n(1-\Omega(\frac{\log k}{k}))}$ when m is "large enough", i.e., $m > n2^{k-1} \ln 2$. When $m \leq n2^{k-1} \ln 2$, we observe that the much simpler approach of merely randomly sampling assignments already works, whp! This is because by Corollary 1 (in the Preliminaries), the number of solutions of $F \in_r P(n,k,m)$ for $m \leq n2^{k-1} \ln 2$ is at least $2^{n/2}2^{-O(nk/2^k)}$ whp. When this event happens, randomly sampling poly(n)$2^{n/2}2^{O(nk/2^k)}$ assignments will uncover a solution whp.

4 Reducing from Random *k*-SAT to Planted Random *k*-SAT

In this section we observe a reduction from random k-SAT to planted k-SAT, which yields the desired algorithm for random k-SAT (see Theorem 3). The following lemma is similar to results in Achlioptas [1], and we present it here for completeness.

Lemma 7 ([1]). *Suppose there exists an algorithm A for planted k-SAT $P(n,k,m)$, for some $m \geq 2^k \ln 2(1-f(k)/2)n$, which finds a solution in time $2^{n(1-f(k))}$ and with probability $1-2^{-nf(k)}$, where $1/k < f(k) = o_k(1)$[2]. Then given a random k-SAT instance sampled from $R^+(n,k,m)$, we can find a satisfiable solution in $2^{n(1-\Omega(f(k)))}$ time with $1-2^{-n\Omega(f(k))}$ probability.*

Proof. Let F be sampled from $R^+(n,k,m)$, and let Z denote its number of solutions with s its expected value. As $f(k) > 1/k$ and $m \geq 2^k \ln 2(1-f(k)/2)n$, Lemmas 3 and 5 together imply that $s \leq 2 \cdot 2^{nf(k)/2}$.

We will now run Algorithm A. Note that if Algorithm A gives a solution it is correct hence we can only have error when the formula is satisfiable but algorithm A does not return a solution. We will now upper bound the probability of A making an error.

$$
\begin{aligned}
&\Pr_{F\in R^+(n,k,m),A}[A \text{ returns no solution}] \\
\leq &\sum_{\sigma\in\{0,1\}^n} \Pr_{F\in R^+(n,k,m),A}[\sigma \text{ satisfies } F \text{ but } A \text{ returns no solution}] \\
\leq &\sum_{\sigma\in\{0,1\}^n} \Pr_{F\in R^+(n,k,m),A}[A \text{ returns no sol} \mid \sigma \text{ satisfies F}] \Pr_{F\in R^+(n,k,m)}[\sigma \text{ satisfies F}] \\
\leq &\sum_{\sigma\in\{0,1\}^n} \Pr_{F\in P(n,k,m,\sigma),A}[A \text{ returns no solution}] \Pr_{F\in R^+(n,k,m)}[\sigma \text{ satisfies F}]
\end{aligned}
$$

where the last inequality used the fact that $R^+(n,k,m)$ conditioned on having σ as a solution is the distribution $P(n,k,m,\sigma)$. Now note that

[2] Note we can assume wlog that $f(k) > 1/k$, as we already have a $2^{n(1-1/k)}$ algorithm for worst-case k-SAT.

$\Pr_{F\in R^+(n,k,m)}[\sigma \text{ satisfies F}] = s/2^n$, and $P(n,k,m) = P(n,k,m,\sigma)$, where σ is sampled uniformly from $\{0,1\}^n$. Hence the expression simplifies to

$$\frac{s}{2^n}\left(2^n \Pr_{F\in P(n,k,m),A}[A \text{ does not return a solution}]\right)$$
$$= s \Pr_{F\in P(n,k,m),A}[A \text{ does not return a solution}].$$

Since $s \leq 2\cdot 2^{nf(k)/2}$, the error probability is $\leq 2\cdot 2^{nf(k)/2}2^{-nf(k)} \leq 2\cdot 2^{-nf(k)/2} = 2^{-\Omega(nf(k))}$. □

Next, we give another reduction from random k-SAT to planted k-SAT. This theorem is different from Lemma 7, in that, given a planted k-SAT algorithm that works in a certain regime of m, it implies a random k-SAT algorithm for *all* values of m.

Reminder of Theorem 2. *Suppose there is an algorithm A for planted k-SAT $P(n,k,m)$, for all $m \geq 2^k \ln 2(1-f(k)/2)n$, which finds a solution in time $2^{n(1-f(k))}$ and with probability $1-2^{-nf(k)}$, where $1/k < f(k) = o_k(1)$. Then for any m', given a random k-SAT instance sampled from $R^+(n,k,m')$, a satisfying assignment can be found in $2^{n(1-\Omega(f(k)))}$ time whp.*

Proof. Let F be sampled from $R^+(n,k,m)$, and let Z denote its number of solutions with s its expected value. The expected number of solutions of F' sampled from $R(n,k,m')$ serves as a lower bound for s. Hence if $m' \leq 2^k \ln 2(1-f(k)/2)n \leq \alpha_d n$, then $s > 2^{nf(k)/2}$ and furthermore, as we have $f(k) > 1/k$, Lemma 3 implies that, $\lim_{n\to\infty} \Pr[Z < s/2^{O(nk/2^k)}] = 0$. So if we randomly sample $O(2^n 2^{O(nk/2^k)}/s) \leq 2^{n(1-\Omega(f(k)))}$ assignments, one of them will satisfy F whp. Otherwise if $m' \geq 2^k \ln 2(1-f(k)/2)n$ then we can use Lemma 7 to solve it in required time. □

Finally, we combine Algorithm 1 for planted k-SAT and the reduction in Theorem 2 to obtain an algorithm for finding solutions of random k-SAT (conditioned on satisfiability). This disproves Super-SETH for random k-SAT.

Reminder of Theorem 3. *Given a random k-SAT instance F sampled from $R^+(n,k,m)$ we can find a solution in $2^{n(1-\Omega(\frac{\log k}{k}))}$ time whp.*

Proof. By Theorem 1 we have an algorithm for planted k-SAT running in $2^{n(1-\Omega(\frac{\log k}{k}))}$ time with $1-2^{-\Omega\left(n\left(\frac{\log k}{k}\right)\right)}$ probability for all $m > (2^{k-1}\ln 2)n$. This algorithm satisfies the required conditions in Theorem 2 with $f(k) = \Omega(\log k/k)$ for large enough k. The implication in Theorem 2 proves the required statement. □

Just as in the case of planted k-SAT, when $m < n(2^k \ln 2 - k)$ we can find solutions of $R^+(n,k,m)$ whp, by merely random sampling assignments. The correctness of random sampling follows from Lemma 3.

5 k-SAT and Unique k-SAT

In this section we give a randomized reduction from k-SAT to Unique k-SAT which implies their equivalence for Super Strong ETH:

Reminder of Theorem 4. *If Unique k-SAT is solvable in $2^{(1-f(k)/k)n}$ time for some unbounded $f(k)$, then k-SAT is solvable in $2^{(1-f(k)/k+O((\log f(k))/k))n}$ time.*

We start with a slight modification of the Valiant-Vazirani lemma.

Lemma 8 (Weighted-Valiant-Vazirani). *Let $S \subseteq \{0,1\}^k = R$ be a set of assignments on variables $x_1, x_2, \ldots x_k$, with $2^{j-1} \leq |S| < 2^j$. Suppose that for each $s \in S$ there exists a* weight $w_s \in \mathbb{Z}^+$*, and let $\bar{w}$ denote the average weight over all $s \in S$. There is a randomized polytime algorithm* **Weighted-Valiant-Vazirani** *that on input (R, j) outputs a matrix $A \in \mathbb{F}_2^{j \times n}$ and a vector $b \in \mathbb{F}_2^j$ such that*

$$\Pr_{A,b}\left[|\{x \mid Ax = b \wedge x \in S\}| = 1, w_s \leq 2\bar{w}\right] > \frac{1}{16}.$$

If the condition in the probability expression is satisfied, we say **Weighted-Valiant-Vazirani** *on (R, j) has succeeded.*

Proof. The original Valiant-Vazirani Lemma [22] gives a randomized polytime algorithm to generate A, b such that for all $s \in S$, $\Pr_{A,b}[\{s\} = \{x \mid Ax = b \wedge x \in S\}] > \frac{1}{8|S|}$. Moreover, by Markov's inequality, we have $\Pr_{s \in S}[w_s \leq 2\bar{w}] \geq 1/2$. Hence the set of $s \in S$ with $w_s \leq 2\bar{w}$ has size at least $|S|/2$. This implies $\Pr_{A,b}[\exists s, \{s\} = \{x \mid Ax = b \wedge x \in S\}, w_s \leq 2\bar{w}] > \left(\frac{1}{8|S|}\right)\left(\frac{|S|}{2}\right) = \frac{1}{16}$. □

Proof of Theorem 4. Let A be an algorithm for Unique k-SAT which runs in time $2^{(1-f(k)/k)n}$.

Let S be the set of SAT assignments to F. Suppose $|S| \geq 2^{nf(k)/k}n$. Then the probability that the random search in lines 1 to 4 never finds a solution is

$$(1 - n2^{nf(k)/k}/2^n)^{2^{n(1-f(k)/k)}} \leq e^{-n}.$$

Thus if $|S| \geq 2^{nf(k)/k}n$ then the algorithm finds a solution whp. From now on, we assume $|S| < 2^{nf(k)/k}n$.

In line 6, we define a sequence of probabilities $p_1, p_2, \ldots, p_k$. Note that

$$\begin{aligned}\sum_{i=1}^{k} p_i &= \sum_{i=1}^{f(k)} p_i + \sum_{i=f(k)+1}^{k} p_i \leq 1/2 + 1/(2f(k)) \sum_{j=1}^{\infty} (1/2f(k))^{j/f(k)} \\ &\leq \frac{1}{2} + \frac{1}{f(k)(1-(1/2f(k))^{1/f(k)})} \leq 1,\end{aligned}$$

as $f(k)$ is unbounded, and $\lim_{x \to \infty} x(1-(1/2x)^{1/x}) = \infty$.

Algorithm 2. Algorithm for k-SAT.

Input: k-SAT formula F

We assume that there is an algorithm A for Unique k-SAT running in time $2^{n(1-f(k)/k)}$.

1: **for** $i = 0$ to $2^{n(1-f(k)/k)}$ **do**
2: sample random solution s
3: **if** s satisfies F **then**
4: Return s
5: Divide n variables into n/k equal parts $R_1, R_2 \ldots R_{n/k}$ and let x^i denote the variables in set R_i
6: Define $p = p_1 = p_2 \ldots = p_{f(k)} = 1/(2f(k))$ and $p_j = p^{j/f(k)}$ for $f(k) \leq j \leq k$
7: $F_0 = F$
8: **for** $u = 1$ to $2^{cn \log(f(k))/k}$ **do**
9: **for** $i = 1$ to n/k **do**
10: Sample z_i from $[k]$ choosing $z_i = j$ with probability p_j
11: $(A_i, b_i) =$ **Weighted-Valiant-Vazirani**(R_i, z_i)
12: $F_i = F_{i-1} \wedge (A_i x^i = b_i)$
13: $s = A(F_{n/k})$
14: Return s if it satisfies F
15: Return unsatisfiable

We will now analyze the i^{th} run of the loop from line 9 to line 14. Let $S_0 = S$, and let S_i be the set of solutions to the formula F_i defined in line 12.

Let E_i be the event that:

1. $2^{z_i - 1} \leq |\{s_{|R_i} \mid s \in S_{i-1}\}| < 2^{z_i}$. [As defined in line 10]
2. for all $s \in S_i$, the restriction on R_i is the same, i.e., $|\{s_{|R_i} \mid s \in S_i\}| = 1$.
3. $|S_{i-1}|/|S_i| \geq 2^{z_i - 2}$, $|S_i| \neq 0$.

In Line 11 we apply **Weighted-Valiant-Vazirani** to (R_i, z_i) with the set of assignments being $\{s_{|R_i} \mid s \in S_{i-1}\}$ where an assignment v has weight $w_v = |\{v = s_{|R_i} \mid s \in S_{i-1}\}|$. For **Weighted-Valiant-Vazirani** to apply, we require that z_i indeed represents an estimate of number of possible assignments to variables of R_i in a satisfying assignment i.e. $2^{z_i - 1} \leq |\{s_{|R_i} \mid s \in S_{i-1}\}| < 2^{z_i}$ which is exactly the condition 1 in E_i. If the call to **Weighted-Valiant-Vazirani** succeeds, then we have that only a unique assignment to R_i remains, i.e, $|\{s_{|R_i} \mid s \in S_i\}| = 1$ which is the condition 2 of E_i. Similarly condition 3 is also implied by the success of **Weighted-Valiant-Vazirani**.

Let y_i satisfy $2^{y_i - 1} \leq |\{s_{|R_i} \mid s \in S_{i-1}\}| < 2^{y_i}$. Then for E_i to be true we need that the sample z_i is equal to y_i, and **Weighted-Valiant-Vazirani** on (R_i, z_i) succeeds.

Let $E = \bigcap_i E_i$. If event E occurs, then the restrictions of all solutions on each R_i's are the same, and there is a solution as $|S_{n/k}| \neq 0$, hence there is a unique satisfying assignment. We wish to lower bound the probability of E occurring.

$$\begin{aligned}\Pr[E] &= \prod_i \Pr[E_i \mid \bigwedge_{j<i} E_j] \\ &\geq \prod_i \Pr[z_i = y_i \mid \bigwedge_{j<i} E_j] \cdot \prod_i \Pr[\mathbf{WVV}(R_i, z_i) \mid \forall j < i, E_j] \\ &\geq \prod_i p_{y_i} \prod_i \left(\frac{1}{16}\right) \qquad \text{[By Lemma 8]} \\ &\geq 16^{-n/k} \prod_i p_{y_i}\end{aligned} \tag{1}$$

When E holds, $|S| = |S_0| = \prod_i |S_{i-1}|/|S_i|$, as $|S_{n/k}| = 1$, Furthermore $\prod_i |S_{i-1}|/|S_i| \geq \prod_i 2^{y_i - 2}$, by condition 3. Since $|S| \leq 2^{nf(k)/k}n$, we have $\prod_i 2^{y_i-2} \leq 2^{nf(k)/k}n$. Taking logarithms, $\sum_i y_i \leq O(n/k) + nf(k)/k \leq O(nf(k)/k)$. Therefore

$$\begin{aligned}\Pr[E] &\geq 16^{-n/k} \prod_i p_{y_i} \text{[Restating equation (1)]} \\ &\geq 16^{-n/k} \prod_{y_i \leq f(k)} p_{y_i} \prod_{y_i > f(k)} p_{y_i} \\ &\geq 16^{-n/k} \cdot (1/2f(k))^{n/k} \cdot \prod_{y_i > f(k)} (1/2f(k))^{(y_i/f(k))} \\ &\geq 16^{-n/k} \cdot (1/2f(k))^{n/k} \cdot (1/2f(k))^{\sum_{y_i > f(k)} (y_i/f(k))} \\ &\geq 16^{-n/k} \cdot (1/2f(k))^{n/k} \cdot (1/2f(k))^{O(n/k)} \\ &\geq 16^{-n/k} \cdot 2^{-O(n \log f(k)/k)} \geq 2^{-O(n \log f(k)/k)}.\end{aligned} \tag{2}$$

As mentioned earlier, if E occurs, then there is a unique SAT assignment and it is found by our Unique k-SAT algorithm A. The probability E does not happen over all $2^{cn(\log f(k))/k}$ runs of the loop on line 8 is at most $1 - 2^{-O(n(\log f(k))/k)})^{2^{cn(\log f(k))/k}} \ll 2^{-n}$, for sufficiently large c. The total running time is $2^{n(1-f(k)/k)} + 2^{cn(\log f(k))/k} \cdot 2^{(1-f(k)/k)n} \leq 2^{(1-f(k)/k+O((\log f(k))/k))n}$. □

Theorem 4 immediately implies an "ultra fine-grained" equivalence between k-SAT and Unique-k-SAT:

Corollary 2. *Unique k-SAT is in $2^{(1-\omega_k(1/k))n}$ time $\Leftrightarrow$ k-SAT is in $2^{(1-\omega_k(1/k))n}$ time.*

Acknowledgement. We thank Erik Demaine for organizing an open problems session for 6.S078 at MIT, during which some results of this paper were proved.

References

1. Achlioptas, D., Coja-Oghlan, A.: Algorithmic barriers from phase transitions. In: IEEE 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, pp. 793–802. IEEE (2008)

2. Ben-Sasson, E., Bilu, Y., Gutfreund, D.: Finding a randomly planted assignment in a random 3CNF (2002, manuscript)
3. Brakensiek, J., Guruswami, V.: Bridging between 0/1 and linear programming via random walks. CoRR abs/1904.04860 (2019). http://arxiv.org/abs/1904.04860
4. Calabro, C., Impagliazzo, R., Kabanets, V., Paturi, R.: The complexity of unique k-SAT: an isolation lemma for k-CNFs. J. Comput. Syst. Sci. **74**(3), 386–393 (2008)
5. Calabro, C., Impagliazzo, R., Paturi, R.: The complexity of satisfiability of small depth circuits. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917, pp. 75–85. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-11269-0_6
6. Chan, T.M., Williams, R.: 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, pp. 1246–1255 (2016)
7. Chao, M.T., Franco, J.: Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k satisfiability problem. Inf. Sci.: Int. J. **51**(3), 289–314 (1990)
8. Coja-Oghlan, A.: A better algorithm for random k-SAT. SIAM J. Comput. **39**(7), 2823–2864 (2010)
9. Coja-Oghlan, A., Krivelevich, M., Vilenchik, D.: Why almost all k-CNF formulas are easy (2007, manuscript)
10. Cook, S.A., Mitchell, D.G.: Finding hard instances of the satisfiability problem: a survey. In: Satisfiability Problem: Theory and Applications, Proceedings of a DIMACS Workshop, Piscataway, New Jersey, USA, 11–13 March 1996, pp. 1–18 (1996)
11. Ding, J., Sly, A., Sun, N.: Proof of the satisfiability conjecture for large k. In: Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, 14–17 June 2015, pp. 59–68 (2015)
12. Feige, U.: Relations between average case complexity and approximation complexity. In: Proceedings on 34th Annual ACM Symposium on Theory of Computing, 19–21 May 2002, Montréal, Québec, Canada, pp. 534–543 (2002)
13. Hertli, T.: 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold ingeneral. SIAM J. Comput. **43**(2), 718–729 (2014). https://doi.org/10.1137/120868177
14. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. J. Comput. Syst. Sci. **62**(2), 367–375 (2001). https://doi.org/10.1006/jcss.2000.1727
15. Krivelevich, M., Vilenchik, D.: Solving random satisfiable 3CNF formulas in expected polynomial time. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, 22–26 January 2006, pp. 454–463 (2006)
16. Paturi, R., Pudlák, P., Saks, M.E., Zane, F.: An improved exponential-time algorithm for k-SAT. J. ACM **52**(3), 337–364 (2005). https://doi.org/10.1145/1066100.1066101
17. Paturi, R., Pudlák, P., Zane, F.: Satisfiability coding lemma. Chicago J. Theor. Comput. Sci. 1999 (1999). http://cjtcs.cs.uchicago.edu/articles/1999/11/contents.html
18. Pudlák, P., Scheder, D., Talebanfard, N.: Tighter hard instances for PPSZ. In: 44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, 10–14 July 2017, Warsaw, Poland, pp. 85:1–85:13 (2017)
19. Schöning, U.: A probabilistic algorithm for k-SAT and constraint satisfaction problems. In: 40th Annual Symposium on Foundations of Computer Science, FOCS 1999, 17–18 October 1999, New York, NY, USA, pp. 410–414 (1999)

20. Selman, B., Mitchell, D.G., Levesque, H.J.: Generating hard satisfiability problems. Artif. intell. **81**(1–2), 17–29 (1996)
21. Traxler, P.: The time complexity of constraint satisfaction. In: Grohe, M., Niedermeier, R. (eds.) IWPEC 2008. LNCS, vol. 5018, pp. 190–201. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79723-4_18
22. Valiant, L.G., Vazirani, V.V.: NP is as easy as detecting unique solutions. Theor. Comput. Sci. **47**(3), 85–93 (1986). https://doi.org/10.1016/0304-3975(86)90135-0
23. Vilenchik, D.: Personal communication
24. Williams, R.: Circuit analysis algorithms. Talk at Simons Institute for Theory of Computing, August 2015. https://youtu.be/adJvi7tL-qM?t=925