On Super Strong ETH

Nikhil Vyas
MIT
nikhilv@mit.edu

Ryan Williams
MIT
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, Pudlak, and Zane (1998).
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 $\varepsilon > 0$.

2012 ACM Subject Classification General and reference → General literature; General and reference

Keywords and phrases Dummy keyword

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Nikhil Vyas*: Supported by an Akamai Presidential Fellowship and NSF CCF-1741615.
Ryan Williams: Supported by NSF CCF-1741615.

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. The inability to find algorithms led researchers to the following two popular hypotheses which strengthen $\mathsf{P} \neq \mathsf{NP}$:

- **Exponential Time Hypothesis (ETH)** [13] There is an $\alpha > 0$ such that no 3-SAT algorithm runs in $2^{\alpha n}$ time.
- **Strong Exponential Time Hypothesis (SETH)** [4] There does not exist a constant $\epsilon > 0$ such that for all k, k-SAT can be solved in $(2-\epsilon)^n$ time.

In fact, the situation for k-SAT algorithms is even worse. The current best known algorithms for k-SAT all have running time $2^{n\left(1-\Omega\left(\frac{1}{k}\right)\right)}$, i.e., time $2^{n\left(1-\frac{c}{k}\right)}$ for some constant $c > 0$. This bound is

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

achieved by multiple paradigms, such as randomized backtracking [16, 15], local search [18], and the polynomial method [5]. Even for simpler variants such as unique-k-SAT, no faster algorithms are known. Hence it is possible that this runtime of $2^{n\left(1-\Omega\left(\frac{1}{k}\right)\right)}$ is actually optimal. This was termed the Super-Strong ETH in a 2015 talk by the second author [23].

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

In this paper, we study Super-SETH in two natural restricted scenarios:

- **Random/Planted k-SAT**. There are two cases generally studied: (a) finding a solution of a random k-SAT instance where each clause is drawn uniformly and independently from the set of all possible k-width clauses, and (b) finding solutions of a planted k-SAT instance, where a random (hidden) solution σ is sampled, then each clause is drawn uniformly and independently from the set of all possible clauses of width k 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 [9, 19] 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 [21]. 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, 15] have an easier analysis when restricted to the case of Unique-k-SAT. PPSZ [15], 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 [6] 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 [7] 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 [14] and random k-SAT (conditioned on satisfiability) by Coja-Oghlan, Krivelevich and Vilenchik [8]. However, both of these results require that m is at least $4^k n/k$ [22]. 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 [21] 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. To address this, Calabro, Impagliazzo, Kabanets and Paturi [3] 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 [20] to $2^{-O(n\log(k)/k)}$. Both of these reductions imply that k-SAT and either both have $2^{\delta n}$ time algorithms for some *universal* $\delta > 0$, or neither of them do (i.e., SETH and the SETH for Unique-k-SAT are equivalen). However these results are not sufficient for an equivalence w.r.t. Super-SETH: for example, it is still possible that k-SAT has no $2^{n(1-\omega(1/k))}$ time algorithms, while Unique-k-SAT has a $2^{n(1-\Omega(\log k/k))}$ time algorithm.

1.2 Our Results

1.2.1 Average-Case k-SAT Algorithms

First we present an algorithm which breaks 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 [16].

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 [2, 1].

▶ **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 with $1 - 2^{-n\Omega(f(k))}$ probability.*

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.*

---

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

▶ Remark 4. 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 [11] conjectures that there are no efficient *refutations* for random k-SAT near the threshold, i.e., there are no 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 is an intriguing open problem.

Our running time of $2^{n(1-\Omega(\frac{\log k}{k}))}$ implies 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 also false for worst-case k-SAT.

For the PPZ algorithm, time lower bounds of the form $2^{n(1-O(\frac{1}{k}))}$ are known [17]. 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 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 [17], 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.

In Appendix A, 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), when m is large.

1.2.2 Unique k-SAT Equivalence

In Section 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 5.** *A $2^{(1-f(k)/k)n}$ time algorithm for Unique k-SAT where $f(k) = \omega_k(1)$ implies a $2^{(1-f(k)/k+O((\log f(k))/k))n}$ algorithm for k-SAT.*

As mentioned earlier, the current best algorithm for k-SAT PPSZ [15] 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 [12]. Theorem 5 implies that, in order to obtain faster algorithms for k-SAT which break Super-Strong ETH, it is 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 a large enough constant. We will 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 for k that can grow unboundedly.

We often 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 the following three distributions for 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 can run it on a random instance from R and return 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 [10].

▶ **Lemma 6** ([10]). *There is a constant k_0 such that for all $k > k_0$, for $\alpha_{sat} = 2^k \ln 2 - \Theta(1)$ 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. This was proven by Achlioptas [1].

▶ **Lemma 7** (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:

▶ **Lemma 8.** *For $F \in_r R^+(n,k,m)$ let Z denote the number of solutions of F. Then for every constant $\delta > 0$, if $m < (1-\epsilon)\alpha_{sat}$ for some constant $\epsilon > 0$, then $2^n(1-\frac{1}{2^k})^m \le E[Z] \le (1+\delta)2^n(1-\frac{1}{2^k})^m$. Furthermore, for $\alpha_d = 2^k \ln 2 - k$, and $m < \alpha_d n$ we have*

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

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, then $E[Z'] = (1-p_n)E[Z]$. By Lemma 6 $\lim_{n\to\infty} p_n \to 0$, hence $2^n(1-\frac{1}{2^k})^m \le E[Z] \le (1+\delta)2^n(1-\frac{1}{2^k})^m$.

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 7, $\Pr[Z' < E[Z']/2^{O(nk/2^k)}]$ tends to 0. ◀

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

▶ **Lemma 9** (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 10.** *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 8. Lemma 9 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 $\lim_{n\to\infty} \Pr[Z' < E[Z]/2^{O(nk/2^k)}] = 0$. ◀

Note that so far, our lemmas regarding the number of solutions do not apply when $m > \alpha_{sat} n$. Next we prove a lemma bounding the number of expected solutions when $m > \alpha_{sat} n$; this may be of independent interest.

▶ **Lemma 11.** *The expected number of solutions of $F \in_r R^+(n,k,m)$ and $F' \in_r P(n,k,m)$ for $m \geq (\alpha_{sat} - 1)n$ is at most $2^{O(n/2^k)}$.*

Proof. Lemma 9 shows that the planted k-SAT distribution $P(n,k,m)$ is biased toward satisfiable formulas with more solutions. Hence the expected number of solutions of $F' \in_r P(n,k,m)$ upper bounds the expected number of solutions of $F \in_r R^+(n,k,m)$. So it suffices for us to upper bound the expected number of solutions of F'.

Let Z denote 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 unsatisfying 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 $E[Z]$.

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

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

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

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

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

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

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 $f''(p) = \frac{-1}{p(1-p)} + k(k-1)(1-p)^{k-2}$. Observe that $f''(p) = 0 \iff p(1-p)^{k-1} = \frac{1}{k(k-1)}$. 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 the following stronger claim: For any σ, if F was sampled from $P(n,k,m,\sigma)$, then we can find a set of $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$ assignments in $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$ time, and with probability $1 - 2^{-\Omega\left(n\left(\frac{\log k}{k}\right)\right)}$ one of them will be σ (the probability is over the planted k-SAT distribution and the randomness of the algorithm). Theorem 1 implies 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 slightly modified version of the PPZ algorithm [16], 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 exists a unit clause then
            set the variable in it to make it true
        else if x_i is unassigned then
            Set x_i randomly.
            i ← i + 1
        else
            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.

We will start with a few useful definitions.

▶ **Definition 12** ([16]). *A clause C is* critical *with respect to variable x and a satisfying assignment σ if x is the only variable in C whose corresponding literal is satisfied by σ.*

▶ **Definition 13.** *A variable x_i in F is* good *for an assignment σ if there exists a clause C in F which is critical with respect to x 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 14.** *A planted k-SAT instance sampled from $P(n, k, m, \sigma)$ with $m > n2^{k-1}\ln 2$ has at least $\Omega(n \log k/k)$ good variables with probability $1 - 2^{-\Omega\left(\frac{n \log k}{k}\right)}$ with respect to the assignment σ.*

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 $|L_g| \le n \ln k/(500k)$. Our goal is to prove a strong upper bound on the probability that E occurs. For all $x_i \in L$, we have that $i \ge n(1 - \ln k/(2k))$. 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

$$\Pr_{C \text{ which satisfies } \sigma}[x_i \in L \text{ is good with respect to } C]$$

$$= \frac{\text{number of clauses for which } x_i \in L \text{ is good}}{\text{total number of clauses satisfying } \sigma} = \frac{\binom{i-1}{k-1}}{\binom{n}{k}(2^k - 1)}$$

$$\ge \frac{1}{2}\left(\frac{i}{n}\right)^{k-1} \frac{k}{2^k n} \quad [\text{As } i \ge n(1 - \ln k/(2k))]$$

$$\geq \frac{1}{2}\left(\frac{i}{n}\right)^k \frac{k}{2^k n}$$
$$\geq \frac{1}{2}\left(1-\frac{\ln k}{2k}\right)^k \frac{k}{2^k n} \quad \text{[As } i \geq n(1-\ln k/(2k))\text{]}$$
$$\geq \frac{1}{2}\left(e^{-\ln k/k}\right)^k \frac{k}{2^k n} \quad \text{[As } k \text{ is large enough, and } e^{-w} \leq 1-w/2 \text{ for small enough } w>0\text{]}$$
$$\geq \frac{1}{2^{k+1} n}$$

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:

$$\Pr[E] \leq \sum_{i=1}^{n \ln k/(500k)} \Pr[\text{exactly } i \text{ variables among the last } n \ln k/(2k) \text{ variables are good}]$$
$$\leq \sum_{i=1}^{n \ln k/(500k)} \binom{n \ln k/(2k)}{i} \left(1-\frac{\ln k}{k2^{k+3}}\right)^m$$
$$\leq n \binom{n \ln k/(2k)}{n \ln k/(500k)} \left(1-\frac{\ln k}{k2^{k+3}}\right)^{n2^{k-1}\ln 2} \quad \text{[As } m > n2^{k-1}\ln 2\text{]}$$
$$\leq n \binom{n \ln k/(2k)}{n \ln k/(500k)} \left(e^{-\frac{\ln k}{k2^{k+3}}}\right)^{n2^{k-1}\ln 2} \quad \text{[As } 1-x \leq e^{-x} \text{ for } x>0\text{]}$$
$$\leq n \binom{n \ln k/(2k)}{n \ln k/(500k)} \left(2^{-\frac{n \ln k}{16k}}\right)$$
$$\leq 2^{-\delta \frac{n \ln k}{k}}$$

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 14, we know that with probability $\geq (1-p)$ 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., $m > n2^{k-1}\ln 2$. For $m \leq n2^{k-1}\ln 2$, we observe that the much simpler approach of randomly sampling assignments works, whp! This is because by Corollary 10 (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. As m decreases further, this sampling approach gives even faster algorithms for finding a solution.

4 Reductions 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 15** ([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 $m \geq 2^k \ln 2(1-f(k)/2)n$, Lemma 8 and 11 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.

$$
\begin{aligned}
&\Pr_{F \in R^+(n,k,m),A}[\text{A does not return a solution}] \\
&\leq \sum_{\sigma \in \{0,1\}^n} \Pr_{F \in R^+(n,k,m),A}[\sigma \text{ satisfies F but } A \text{ does not return a solution}] \\
&\leq \sum_{\sigma \in \{0,1\}^n} \Pr_{F \in R^+(n,k,m),A}[A \text{ does not return a solution} \mid \sigma \text{ satisfies F}] \Pr_{F \in R^+(n,k,m)}[\sigma \text{ satisfies F}] \\
&\leq \sum_{\sigma \in \{0,1\}^n} \Pr_{F \in P(n,k,m,\sigma),A}[A \text{ does not return a solution}] \Pr_{F \in R^+(n,k,m)}[\sigma \text{ satisfies F}]
\end{aligned}
$$

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

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

As $s \leq 2 \cdot 2^{nf(k)/2}$ 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 the previous one, 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 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. The expected number of solutions for 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 8 implies that, $\lim_{n\to\infty} \Pr[Z < s/2^{O(nk/2^k)}] = 0$. Hence, if we randomly sample $O(2^n 2^{O(nk/2^k)}/s) = 2^{n(1-\Omega(f(k)))}$ 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 15 to solve it in required time. ◀

[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.

Now we combine Algorithm 1 for planted k-SAT and the reduction in Theorem 2, to get 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 8.

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 5. A $2^{(1-f(k)/k)n}$ time algorithm for Unique k-SAT where $f(k)=\omega_k(1)$ implies a $2^{(1-f(k)/k+O((\log f(k))/k))n}$ time randomized algorithm for finding a solution of k-SAT.

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

▶ **Lemma 16** (Weighted-Valiant-Vazirani). *Let $S\subseteq\{0,1\}^k=R$ be a set of assignments on variables $x_1,x_2,\dots x_k$, with $2^{j-1}\leq|S|<2^j$. Suppose for each $s\in S$ we are also given a* weight *$w_s\in\mathbb{Z}^+$, and let $\bar{w}$ denote the average weight over all $s\in S$. Then there is a randomized polytime algorithm to guess 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 [21] gives a randomized polytime algorithm to guess 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 5. Let A be an algorithm for Unique k-SAT which runs in time $2^{(1-f(k)/k)n}$.

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 $R_1, R_2 \dots R_{n/k}$
6: Define $p = p_1 = p_2 \dots = p_{f(k)} = 1/2f(k)$ and $p_j = p^{j/f(k)}$ for $f(k) \le j \le k$
7: **for** $u = 1$ to $2^{cn\log(f(k))/k}$ **do**
8: **for** $i = 1$ to n/k **do**
9: Sample z_i from $[k]$ choosing $z_i = j$ with probability p_j
10: $F' =$ Weighted-Valiant-Vazirani(R_i, z)
11: $s = A(F')$
12: Return s if it satisfies F
13: Return unsatisfiable

Let S be the set of solutions of the k-SAT instance F. Suppose F has at least $2^{nf(k)/k}n$ solutions, i.e., $|S| \ge 2^{nf(k)/k}n$. Then the probability that the random search in lines 1 to 4 never finds a solution is $(1 - n2^{nf(k)/k}/2^n)^{2^{n(1-f(k)/k)}} \le e^{-n}$. Thus if $|S| \ge 2^{nf(k)/k}n$ 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 $p_1, p_2, \dots, p_k$. Note that $\sum_{i=1}^{k} p_i = \sum_{i=1}^{f(k)} p_i + \sum_{i=f(k)+1}^{k} p_i \le 1/2 + (1/2f(k)) \sum_{j=1}^{\infty} (1/2f(k))^{j/f(k)} \le \frac{1}{2} + \frac{1}{f(k)(1-(1/2f(k))^{1/f(k)})} \le 1$, as $f(k) = \omega_k(1)$ and $\lim_{x\to\infty} x(1 - (1/2x)^{1/x}) = \infty$.

We will now analyze one run of the loop from line 8 to line 12. Let S_i be the set of solutions after applying constraints on R_1 to R_i, where $S_0 = S$ is the initial set of solutions. Let E_i be the event that

1. $2^{z_i - 1} \le |\{s_{|R_i} \mid s \in S_{i-1}\}| < 2^{z_i}$. [As defined in line 9]
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. $|S_{i-1}|/|S_i| \ge 2^{z_i - 2}$, $|S_i| \ne 0$.

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

Let y_i satisfy $2^{y_i - 1} \le |\{s_{|R_i} \mid s \in S_{i-1}\}| < 2^{y_i}$. Then for condition 1 to be satisfied we need that the sample z_i be equal to y_i. For conditions 2 and 3 to be satisfied we only need that Weighted-Valiant-Vazirani (WVV) on Line 10 to succeed on (R_i, z_i) as described in Lemma 16.

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

When E holds, we have $|S| = |S_0| = \prod_i |S_{i-1}|/|S_i|$, as $|S_{n/k}| = 1$. Furthermore $\prod |S_{i-1}|/|S_i| \ge \prod_i 2^{y_i - 2}$, by condition 3. By the initial random sampling, we have $|S| \le 2^{nf(k)/k}n$. Hence

$\prod_i 2^{y_i-2} \leq 2^{nf(k)/k}n$ which implies that $\sum_i y_i \leq O(n/k) + nf(k)/k \leq O(nf(k)/k)$. Therefore

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

As mentioned earlier, if E occurs, then there is a unique SAT assignment, which will be found by the Unique k-SAT algorithm A. The probability that E does not happen over all $2^{cn(\log f(k))/k}$ runs of the loop on line 7 is at most $(1-2^{-O(n(\log f(k))/k)})^{2^{cn(\log f(k))/k}} \ll 2^{-n}$, for a large enough constant c. The total running time is $2^{n(1-f(k)/k)} + 2^{cn(\log f(k))/k} 2^{(1-f(k)/k)n} = 2^{(1-f(k)/k+O((\log f(k))/k))n}$. ◀

The reduction above immediately implies that Super-SETH is equivalent for k-SAT and Unique-k-SAT.

▶ **Corollary 17.** *A $2^{(1-\omega_k(1/k))n}$ time algorithm for Unique k-SAT implies a $2^{(1-\omega_k(1/k))n}$ algorithm for k-SAT.*

Say that a (Unique)k-SAT algorithm has *advantage* δ if it runs in $2^{n(1-\delta)}$ time. Let $g(k)$ be the advantage of the best k-SAT algorithm, and let $g_u(k)$ be the advantage of the best Unique-k-SAT algorithm. As mentioned earlier, current algorithms lower bound both $g(k)$ and $g_u(k)$ by $\Omega(1/k)$. Our reduction shows that these advantages are asympotically identical if Super-Strong ETH is false:

▶ **Corollary 18.** *If $g_u(k) = \omega_k(1/k)$, then $\lim_{k\to\infty} \frac{g_u(k)}{g(k)} = 1$.*

References

1 Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 793–802. IEEE, 2008.

2 Eli Ben-Sasson, Yonatan Bilu, and Danny Gutfreund. Finding a randomly planted assignment in a random 3cnf. *manuscript*, 2002.

3 Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-sat: An isolation lemma for k-cnfs. *Journal of Computer and System Sciences*, 74(3):386–393, 2008.

4 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, pages 75–85, 2009.

5 Timothy M. Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255, 2016.

6 Ming-Te Chao and John Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k satisfiability problem. *Information Sciences: an International Journal*, 51(3):289–314, 1990.

7 Amin Coja-Oghlan. A better algorithm for random k-sat. *SIAM Journal on Computing*, 39(7):2823–2864, 2010.

8 Amin Coja-Oghlan, Michael Krivelevich, and Dan Vilenchik. Why almost all k-cnf formulas are easy. *manuscript*, 2007.

9 Stephen A. Cook and David G. Mitchell. Finding hard instances of the satisfiability problem: A survey. In *Satisfiability Problem: Theory and Applications, Proceedings of a DIMACS Workshop, Piscataway, New Jersey, USA, March 11-13, 1996*, pages 1–18, 1996.

10 Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large k. In *STOC*, pages 59–68, 2015.

11 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 534–543, 2002.

12 Timon Hertli. 3-sat faster and simpler - unique-sat bounds for PPSZ hold in general. *SIAM J. Comput.*, 43(2):718–729, 2014. URL: `https://doi.org/10.1137/120868177`, `doi:10.1137/120868177`.

13 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. URL: `https://doi.org/10.1006/jcss.2000.1727`, `doi:10.1006/jcss.2000.1727`.

14 Michael Krivelevich and Dan Vilenchik. 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, January 22-26, 2006*, pages 454–463, 2006.

15 Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for *k*-sat. *J. ACM*, 52(3):337–364, 2005. URL: `http://doi.acm.org/10.1145/1066100.1066101`, `doi:10.1145/1066100.1066101`.

16 Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chicago J. Theor. Comput. Sci.*, 1999, 1999. URL: `http://cjtcs.cs.uchicago.edu/articles/1999/11/contents.html`.

17 Pavel Pudlák, Dominik Scheder, and Navid Talebanfard. Tighter hard instances for PPSZ. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 85:1–85:13, 2017.

18 Uwe Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 410–414, 1999.

19 Bart Selman, David G Mitchell, and Hector J Levesque. Generating hard satisfiability problems. *Artificial intelligence*, 81(1-2):17–29, 1996.

20 Patrick Traxler. The time complexity of constraint satisfaction. In *Parameterized and Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings*, pages 190–201, 2008.

21 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. URL: `https://doi.org/10.1016/0304-3975(86)90135-0`, `doi:10.1016/0304-3975(86)90135-0`.

22 Dan Vilenchik. personal communication.

23 Ryan Williams. Circuit analysis algorithms. Talk at Simons Institute for Theory of Computing, available at `https://youtu.be/adJvi7tL-qM?t=925`, August 2015.

A Planted and Random k-SAT for large m

In Sections 3 and 4 we gave algorithms for random k-SAT that work at the threshold and for all other values of the clause density. In this section, we work in the regime where the number of clauses m is bounded away from the threshold, and give an improved running time analysis for this case. The proofs follow a similar structure to the proofs in Section 3 and 4. As mentioned before, polynomial-time algorithms finding solutions to random k-SAT instances currently require m to be at least $\frac{4^k}{k}n$. To our knowledge, no improved algorithms were known for $2^k n < m < \frac{4^k}{k}n$ other than the worst case k-SAT algorithms.

▶ **Lemma 19.** *Let $z = (\ln(m/n) - k\ln 2)/k$. A planted k-SAT F instance sampled from $P(n,k,m,\sigma)$ with $2^{k+o(k)}n \geq m \geq 2^k n$ has at least $\Omega(nz)$ good variables, with probability at least $1 - 2^{-\Omega(nz)}$ with respect to the assignment σ.*

Proof. In this proof, by "good/bad variables" we mean "good/bad variables with respect to σ" (see Section 3 to recall the definition of good/bad).

Let $F \in_r P(n,k,m,\sigma)$ and let L be the last (when sorted by index) $nz/2$ variables. Let L_g, L_b be the good and bad variables respectively, with respect to σ, among L. Let E denote the event that $|L_g| \leq nz/500$.

Our goal is to prove a strong upper bound on the probability that E occurs. For any $x_i \in L$, we have that $i \geq n(1-z/2)$. If a clause C is good with respect to $x_i \in L_b$, then we know that C does not occur in F. Next, we will lower bound the probability of such a clause occurring 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 all clauses satisfying σ. We derive:

$$\begin{aligned}
&\Pr[C \text{ is good w.r.t. } x_i \in L] \\
&= \frac{\text{Number of clauses which will make } x_i \in L \text{ good}}{\text{Total number of clauses which satisfy } \sigma} \\
&= \frac{\binom{i-1}{k-1}}{\binom{n}{k}(2^k-1)} \\
&\geq \frac{1}{2}\left(\frac{i}{n}\right)^k \frac{k}{2^k n} \quad [\text{As } i \geq n(1-z/2), z = o(1)] \\
&\geq \frac{1}{2}\left(1 - \frac{z}{2}\right)^k \frac{k}{2^k n} \quad [\text{As } i \geq n(1-z/2)] \\
&\geq \frac{1}{2}\left(e^{-z}\right)^k \frac{k}{2^k n} \quad [\text{As } z = o(1) \text{ and } e^{-w} \leq 1 - w/2 \text{ for small enough } w > 0] \\
&\geq \frac{e^{-zk}}{2^{k+1}n}
\end{aligned}$$

If E is true, then $|L_b| > nz/4$. Therefore, the probability of sampling a clause that makes some variable $x_i \in L_b$ good is at least $\frac{ze^{-zk}}{2^{k+3}}$, as the set of clauses which make different variables good are disjoint. Now we upper bound the probability that E occurs.

$$\begin{aligned}
\Pr[E] &\leq \sum_{i=1}^{nz/500} \Pr[\text{Exactly } i \text{ good variables among the last } nz/2 \text{ variables}] \\
&\leq \sum_{i=1}^{nz/500} \binom{nz/2}{i}\left(1 - \frac{ze^{-zk}}{2^{k+3}}\right)^m \\
&\leq n\binom{nz/2}{nz/500}\left(1 - \frac{ze^{-zk}}{2^{k+3}}\right)^{ne^{zk}2^k} \quad [\text{As } m = e^{zk}2^k n] \\
&\leq n\binom{nz/2}{nz/500}\left(e^{-\frac{ze^{-zk}}{2^{k+3}}}\right)^{ne^{zk}2^k} \quad [\text{As } 1 - x \leq e^{-x} \text{ for } x > 0] \\
&\leq n\binom{nz/2}{nz/500}\left(e^{-\frac{nz}{8}}\right)
\end{aligned}$$

$$\leq 2^{-\delta nz},$$

for appropriately small but constant $\delta > 0$. This proves the lemma statement. ◀

▶ **Theorem 20.** *Given a planted k-SAT instance F sampled from $P(n,k,m)$ with $2^{k+o(k)}n > m > 2^k n$ define $z = (\ln(m/n) - k\ln 2)/k$ and $z' = z + \ln k/k$, we can find a solution of F in $2^{n(1-\Omega(z'))}$ time with at least $1 - 2^{-\Omega(nz')}$ probability (over the planted k-SAT distribution and the randomness of the algorithm).*

Proof. By Lemma 19, we know that with probability at least $1-p$ for $p = 2^{-\Omega(nz)}$, the number of good variables in F (wrt the hidden planted solution σ) is at least γnz for some $\gamma > 0$. For such instances, one run of the PPZ algorithm will output σ with probability at least $2^{-n(1-\gamma z)}$. Repeating the PPZ algorithm for poly(n)$2^{n(1-\gamma z)}$ times implies a success probability of at least $1-p$ for $p' = 2^{-n}$. The overall error probability is at most $p + p' \leq 2^{-\Omega(nz)}$.

Also by Theorem 1, there exists a random k-SAT algorithm running in $2^{n(1-\Omega(\frac{\log k}{k}))}$ time with $1 - 2^{-\Omega\left(n\left(\frac{\log k}{k}\right)\right)}$ success probability. Together, these algorithms imply an algorithm running in $2^{n(1-\Omega(z'))}$ time with $1 - 2^{-\Omega(nz')}$ probability (over the planted k-SAT distribution and the randomness of the algorithm). ◀

▶ **Theorem 21.** *Given a random k-SAT instance F sampled from $R^+(n,k,m)$ with $2^{k+o(k)}n > m > 2^k n$ define $z = (\ln(m/n) - k\ln 2)/k$ and $z' = z + \ln k/k$, we can find a solution of F in $2^{n(1-\Omega(z'))}$ time with $1 - 2^{-\Omega(nz')}$ probability (over the random k-SAT distribution R^+ and the randomness of the algorithm).*

Proof. This follows directly from composing the algorithm in Theorem 20 and the reduction in Lemma 15. ◀

As an example, the above theorem implies: For $F \in_r R^+(n,k,m)$ and $m = 2^{k+\sqrt{k}}n$ we have a $2^{n(1-\Omega(1/\sqrt{k}))}$ algorithm which works with $1 - 2^{-\Omega(n/\sqrt{k})}$ probability.

Next we will increase m even further, and prove there are more good variables for the PPZ algorithm in this case.

▶ **Lemma 22.** *Let $t > 2$ be a constant. Given a planted k-SAT instance F sampled from $P(n,k,m,\sigma)$ with $m \geq t^k n$, F has at least $n(1-2/t)(1-2/k)$ good variables with probability $1 - 2^{-\Omega(n(1-2/t))}$ with respect to the assignment σ.*

Proof. The proof is similar to that of Lemma 19. As before, by "good/bad variables" we mean "good/bad variables with respect to the assignment σ".

Let $F \in_r P(n,k,m,\sigma)$ and let L be the last (when sorted by index) nz variables where $z = 1 - 2/t$. Let L_g, L_b be the good and bad variables respectively, with respect to σ, among L. Let E be the event that $|L_b| > \gamma nz$, where $\gamma = 2/k$.

As in previous cases, we want to give a strong upper bound on the probability that event E occurs. For any $x_i \in L$, we have that, $i \geq n(1-z)$. If clause C is good with respect to $x_i \in L_b$, then we know C does not occur in F. As before, our next step is to lower bound the probability of such a clause occurring 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 all clauses which satisfy σ. Therefore

$$\Pr[C \text{ is good with respect to } x_i \in L]$$

$$= \frac{\text{Number of clauses which will make } x_i \in L \text{ good}}{\text{Total number of clauses which satisfy } \sigma}$$

$$= \frac{\binom{i-1}{k-1}}{\binom{n}{k}(2^k-1)}$$
$$\geq \frac{1}{2}\left(\frac{i}{n}\right)^k \frac{k}{2^k n} \quad [\text{As } i \geq n(1-z) = \Omega(n)]$$
$$\geq \frac{1}{2}(1-z)^k \frac{k}{2^k n} \quad [\text{As } i \geq n(1-z)]$$
$$= \frac{k(1-z)^k}{2^{k+1} n}$$

If E is true, then $|L_b| > \gamma nz$. So the probability of sampling a clause that makes a variable $x_i \in L_b$ good is at least $\frac{\gamma kz(1-z)^k}{2^{k+1}}$, as the sets of clauses which make different variables good are disjoint sets. Our upper bound on the event E is then calculated as follows:

$$\Pr[E] \leq \sum_{i=1}^{nz(1-\gamma)} \Pr[\text{Exactly } i \text{ good variables among the last } nz \text{ variables}]$$
$$\leq \sum_{i=1}^{nz(1-\gamma)} \binom{nz}{i}\left(1 - \frac{\gamma kz(1-z)^k}{2^{k+1}}\right)^m$$
$$\leq 2^{nz}\left(1 - \frac{\gamma kz(1-z)^k}{2^{k+1}}\right)^{t^k n} \quad [\text{As } m > t^k n]$$
$$\leq 2^{nz}\left(1 - \frac{z(1-z)^k}{2^k}\right)^{t^k n} \quad [\gamma = 2/k]$$
$$\leq 2^{n(1-2/t)}\left(1 - \frac{(1-2/t)2^k}{t^k 2^k}\right)^{t^k n} \quad [\text{Substituting value of } z]$$
$$\leq 2^{n(1-2/t)}\left(1 - \frac{(1-2/t)}{t^k}\right)^{t^k n}$$
$$\leq 2^{n(1-2/t)} e^{-n(1-2/t)} \quad [\text{As } 1-x \leq e^{-x} \text{ for } x > 0]$$
$$\leq 2^{-\delta n(1-2/t)},$$

for appropriately small but constant $\delta > 0$. This proves the lemma statement. ◀

▶ **Theorem 23.** *Given a planted k-SAT instance F sampled from $P(n,k,m)$ with $m \geq t^k n$ where $t > 2$ is a constant, we can find a solution of F in $2^{n(1-(1-2/t)(1-2/k))}$poly(n) time with $1 - 2^{-\Omega(n(1-2/t))}$ probability (over the planted k-SAT distribution and the randomness of the algorithm).*

Proof. By Lemma 22, there is probability at least $1-p$ for $p = 2^{-\Omega(n(1-2/t))}$ that the number of good variables in F is at least $n(1-2/t)(1-2/k)$ with respect to the hidden planted solution σ. For such instances, one run of the PPZ algorithm outputs σ with probability at least $2^{-n(1-(1-2/t)(1-2/k))}$. Repeating PPZ for poly(n)$2^{n(1-(1-2/t)(1-2/k))}$ times implies success probability at least $1-p'$ for $p' = 2^{-n}$. The overall error probability is at most $p + p' \leq 2^{-\Omega(n(1-2/t))}$. ◀

In order to use Theorem 23 to obtain algorithms for R^+, we need a more refined version of Lemma 15.

▶ **Lemma 24.** *Suppose there is an algorithm A for planted k-SAT $P(n,k,m)$ for some $m \geq \alpha_{sat} n$ which finds a solution in time $2^{n(1-f(k))}$ and with probability p. Then, given a random k-SAT instance F sampled from $R^+(n,k,m)$, we can find a solution to F in $2^{n(1-f(k))}$ time with at least $1-(1-p)2^{O(n/2^k)}$ probability.*

Proof. Let F be sampled from $R^+(n,k,m)$, let Z denote the number of solutions, and let s be its expected value. As $m \geq \alpha_{sat} n$, Lemma 11 implies $s \leq 2^{O(n/2^k)}$.

Suppose we simply run Algorithm A. If Algorithm A gives a solution, it is correct, hence our only source of error is when the formula is satisfiable but algorithm A does not return a solution. We can upper bound the probability of A making an error in this way as follows:

$$
\begin{aligned}
&\Pr_{F \in R^+(n,k,m),A}[\text{A does not return a solution}] \\
&\leq \sum_{\sigma \in \{0,1\}^n} \Pr_{F \in R^+(n,k,m),A}[\sigma \text{ satisfies F but } A \text{ does not return a solution}] \\
&\leq \sum_{\sigma \in \{0,1\}^n} \Pr_{F \in R^+(n,k,m),A}[A \text{ does not return a solution} \mid \sigma \text{ satisfies F}] \Pr_{F \in R^+(n,k,m)}[\sigma \text{ satisfies F}] \\
&\leq \sum_{\sigma \in \{0,1\}^n} \Pr_{F \in P(n,k,m,\sigma),A}[A \text{ does not return a solution}] \Pr_{F \in R^+(n,k,m)}[\sigma \text{ satisfies F}],
\end{aligned}
$$

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

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

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

As $s \leq 2^{O(n/2^k)}$, the error probability is at most $2^{O(n/2^k)}(1-p)$. ◀

▶ **Theorem 25.** *Let $t > 2$ be a constant. Given a random k-SAT instance F sampled from $R^+(n,k,m)$ with $m \geq t^k n$, we can find a solution of F in $2^{n(1-(1-2/t)(1-2/k))}$poly(n) time with $1-2^{-\Omega(n(1-2/t))}$ probability (over the planted k-SAT distribution and the randomness of the algorithm).*

Proof. The algorithm in Theorem 23 and the reduction in Lemma 24 imply that we can find a solution of F in $2^{n(1-(1-2/t)(1-2/k))}$poly(n) time with $1-2^{O(n/2^k)}2^{-\Omega(n(1-2/t))} = 1-2^{-\Omega(n(1-2/t))}$ probability. ◀