Results on a Super Strong Exponential Time Hypothesis[*]

Nikhil Vyas,[†] Ryan Williams[‡]
MIT, USA

Abstract

All known SAT-solving paradigms (backtracking, local search, and the polynomial method) only yield a $2^{n(1-1/O(k))}$ time algorithm for solving k-SAT in the worst case, where the big-O constant is independent of k. For this reason, it has been hypothesized that k-SAT cannot be solved in worst-case $2^{n(1-f(k)/k)}$ time, for any unbounded $f : \mathbb{N} \to \mathbb{N}$. This hypothesis has been called the "Super-Strong Exponential Time Hypothesis" (Super Strong ETH), modeled after the ETH and the Strong ETH. We prove two results concerning 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. In particular, 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. It is natural to hypothesize that the worst-case (exponential-time) complexity of Unique k-SAT is substantially less than that of k-SAT. Improving prior reductions, we show the time complexities of Unique k-SAT and k-SAT are very tightly related: 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 $\varepsilon > 0$. Thus, refuting Super Strong ETH in the unique solution case would refute Super Strong ETH in general.

[*]Supported by NSF Grant CCF-1741615 and CCF-1909429.
[†]nikhilv@mit.edu
[‡]rrw@mit.edu

Introduction

k-SAT is the canonical NP-complete problem for $k \geq 3$: *Given a Boolean formula in conjunctive normal form with clauses of width at most k, is it satisfiable?* In practice, k-SAT is often cited as a "solved problem" (Gomes et al. 2008), due to the incredible performance of modern SAT solvers on instances arising from practice (mostly hardware and software verification).

However, it is very possible that in the future, the demands and designs from practice will change significantly, leading to significantly different SAT instances. In general, the *worst-case* complexity of k-SAT is far from understood, in spite of tremendous effort devoted to finding faster worst-case algorithms. Because it is widely believed that $\mathsf{P} \neq \mathsf{NP}$, the search has been confined to super-polynomial-time algorithms. Although is trivial to obtain an algorithm running in $2^n \cdot m^{O(1)}$ time on k-SAT instances with m clauses and n variables, we cannot seem to improve the base of the exponent below 2: there are are no known algorithms for k-SAT which run in $(2-\epsilon)^n \cdot m^{O(1)}$ time for a universal constant $\epsilon > 0$, independent of k. This apparent barrier to algorithms led researchers to the following two popular hypotheses which strengthen $\mathsf{P} \neq \mathsf{NP}$:

- **Exponential Time Hypothesis (ETH)** (Impagliazzo and Paturi 2001) There is an $\alpha > 0$ such that no 3-SAT algorithm runs in $2^{\alpha n}$ time on all n-variable instances.
- **Strong Exponential Time Hypothesis (SETH)** (Calabro, Impagliazzo, and Paturi 2009) There is no $\epsilon > 0$ such that for all k, k-SAT can be solved in $(2-\epsilon)^n$ time.

In fact, the performance of known worst-case k-SAT algorithms is even worse than what the hypotheses conjecture. 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 a constant $c > 0$. As $k \to \infty$, the running time bound converges to 2^n, but moreover this bound converges to 2^n at a certain rate.

Somewhat surprisingly, this best-known $2^{n\left(1-\Omega\left(\frac{1}{k}\right)\right)}$ bound can be achieved by multiple algorithmic paradigms, such as randomized backtracking (Paturi, Pudlák, and Zane 1999; Paturi et al. 2005), local search (Schöning 1999), and the polynomial method (Chan and Williams 2016). Even for simpler variants such as Unique k-SAT (where we are promised there is at most one satisfying assignment), there are no known algorithms with a better dependence on k in the exponent. Hence it is consistent with current theory that this runtime of $2^{n\left(1-\Omega\left(\frac{1}{k}\right)\right)}$ is actually optimal for worst-case k-SAT. This possibility was termed the Super-Strong ETH in a 2015 talk by the second author (Williams August 2015).

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

In the full version of this work (Vyas and Williams 2019), we study Super-SETH in two natural restricted scenarios:

- **Random/Planted k-SAT**. In the case of random k-SAT, two cases are generally studied. The first is that of finding a SAT assignment to a random k-SAT instance, where each clause is drawn uniformly and independently from the set of all possible k-width clauses. The second case is that of finding SAT assignments to planted k-SAT instances, where a random (hidden) solution σ is sampled, then each clause is drawn uniformly and independently from the set of all possible k-width clauses that satisfy σ. Random k-SAT has a well-known "sharp threshold" behavior: depending on the ratio of clauses-to-variables, the formula is either very likely to be satisfiable, or very likely to be unsatisfiable. In particular, 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 $m > (\alpha_{sat} + \epsilon)n$ (Ding, Sly, and Sun 2015). 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 (Cook and Mitchell 1996; Selman, Mitchell, and Levesque 1996) 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**. The Unique k-SAT problem 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 (Valiant and Vazirani 1986). As mentioned earlier, the best known algorithms for Unique k-SAT have the same running time behavior 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 (PPZ and PPSZ) (Paturi, Pudlák, and Zane 1999; Paturi et al. 2005) have an easier analysis when restricted to the case of Unique k-SAT (Paturi et al. 2005; Hertli 2014). The current best known algorithm for k-SAT (Hansen et al. 2019) only mildly improves over the PPSZ algorithm. **Could worst-case algorithms for Unique k-SAT be marginally faster than those for k-SAT?**

In principle, both of the above special cases could have the same exponential-time complexity as k-SAT, or both could be easier, at least from the perspective of Super-SETH (where the dependence on k in the exponent matters). In the full version of this work (Vyas and Williams 2019), 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 essentially the *same* for the two problems.

Prior Work

There has been substantial work on polynomial-time algorithms for random k-SAT that return satisfying assignments when the number of clauses m is noticeably below the threshold. Note that, even though we know that instances below the threshold are *satisfiable* with high probability, that does not immediately give a way to *find* a satisfying assignment. Chao and Franco (Chao and Franco 1990) first proved that the unit clause heuristic (the same key component of the PPZ algorithm (Paturi, Pudlák, and Zane 1999)) finds solutions with high probability for random k-SAT, when $m \leq c2^k n/k$ for some constant $c > 0$. Currently, the best-known polynomial-time algorithm in this regime is by Coja-Oghlan (Coja-Oghlan 2010) and it can find satisfying assignments for random k-SAT (whp) for $m \leq c2^k n \log k/k$ for some constant $c > 0$. Interestingly, polynomial time algorithms are also known for the case of 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 that find satisfying assignments to planted k-SAT instances by Krivelevich and Vilenchik (Krivelevich and Vilenchik 2006) and random k-SAT (conditioned on satisfiability) by Coja-Oghlan, Krivelevich and Vilenchik (Coja-Oghlan, Krivelevich, and Vilenchik 2007). However, both of these results require that m is at least $4^k n/k$ (Vilenchik 2019). To our knowledge, no algorithmic improvements better than $2^{n-n/O(k)}$ time have yet been reported for random k-SAT very close to the threshold.

(Valiant and Vazirani 1986) gave a poly-time randomized reduction from SAT instances F on n variables to Unique-SAT instances F' on n variables such that, if F is SAT, then F' is a unique satisfying assignment with probability at least $\Omega(1/n)$, and if F is UNSAT then F' is UNSAT. This reduction does not apply to convert k-SAT instances to Unique k-SAT instances, as they do not preserve the clause width (nor do they preserve the number of variables, when transformed into k-SAT instances in the natural way). To address this, (Calabro et al. 2008) 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 2008) to $2^{-O(n \log(k)/k)}$. Both reductions imply that either both k-SAT and Unique k-SAT have $2^{\delta n}$ time algorithms for some *universal* $\delta < 1$, or neither of them do. In other words, SETH for k-SAT and the corresponding SETH for Unique k-SAT are equivalent. However, these results are not sufficient for an equivalence with respect to Super-SETH: for example, assuming the above 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.

Our Results

Average-Case k-SAT Algorithms First we present an algorithm breaking 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 satisfying assignment for random-k-SAT (conditioned on satisfiability) whp, for all values of m. In fact, our algorithm is an old one from the SAT algorithms literature: the PPZ algorithm of (Paturi, Pudlák, and Zane 1999).

In order to show that the PPZ algorithm solves random k-SAT faster, 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 on n variables and m clauses 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 randomness in the planted k-SAT distribution, and the randomness of the algorithm).*

Next, we give an efficient 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 (Ben-Sasson, Bilu, and Gutfreund 2002; Achlioptas and Coja-Oghlan 2008).

Theorem 2. *Suppose there is an algorithm A for planted k-SAT on n variables and m clauses, 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.*

Together, the two above theorems yield:

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 (Feige 2002) conjectures that there are no efficient* refutations *for random k-SAT above 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, it is known that there are instances of k-SAT for which PPZ requires a running time of $2^{n(1-O(\frac{1}{k}))}$ (Pudlák, Scheder, and Talebanfard 2017). 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 essentially gives the current best known running time for k-SAT for large k, we only know $2^{n\left(1-O\left(\frac{\log k}{k}\right)\right)}$ lower bounds (Pudlák, Scheder, and Talebanfard 2017), matching our upper bounds for the random case. Hence it is possible that PPSZ (or perhaps its recent improvement (Hansen et al. 2019)) actually runs in $2^{n\left(1-\Omega\left(\frac{\log k}{k}\right)\right)}$ time for worst-case k-SAT.

Unique k-SAT Equivalence We present a "mildly exponential" time reduction from (worst-case) k-SAT to Unique k-SAT, showing that the runtime exponents of the two problems are essentially equivalent:

Theorem 4. *An algorithm running in $2^{(1-f(k)/k)n}$ time for Unique k-SAT (where $f(k)$ is unbounded) implies a $2^{(1-f(k)/k+O((\log f(k))/k))n}$ time algorithm for k-SAT.*

As mentioned earlier, the current best algorithm for k-SAT PPSZ (Paturi et al. 2005) 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 2014). Theorem 4 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.

Conclusion

The proofs of our results can be found in the SAT'19 conference version (Vyas and Williams 2019). Since that paper appeared, another algorithm for random k-SAT has been announced (based on local search) that achieves a running time of $2^{n(1-O(\log^2 k)/k)}$ (Lincoln and Yedidia 2019).

References

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

Ben-Sasson, E.; Bilu, Y.; and Gutfreund, D. 2002. Finding a randomly planted assignment in a random 3cnf. *manuscript*.

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

Calabro, C.; Impagliazzo, R.; and Paturi, R. 2009. 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*, 75–85.

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

Chao, M.-T., and Franco, J. 1990. 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.

Coja-Oghlan, A.; Krivelevich, M.; and Vilenchik, D. 2007. Why almost all k-cnf formulas are easy. *manuscript*.

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

Cook, S. A., and Mitchell, D. G. 1996. 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*, 1–18.

Ding, J.; Sly, A.; and Sun, N. 2015. Proof of the satisfiability conjecture for large k. In *STOC*, 59–68.

Feige, U. 2002. 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*, 534–543.

Gomes, C. P.; Kautz, H. A.; Sabharwal, A.; and Selman, B. 2008. Satisfiability solvers. In *Handbook of Knowledge Representation*. 89–134.

Hansen, T. D.; Kaplan, H.; Zamir, O.; and Zwick, U. 2019. Faster *k*-sat algorithms using biased-ppsz. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, 578–589.

Hertli, T. 2014. 3-sat faster and simpler - unique-sat bounds for PPSZ hold in general. *SIAM J. Comput.* 43(2):718–729.

Impagliazzo, R., and Paturi, R. 2001. On the complexity of k-sat. *J. Comput. Syst. Sci.* 62(2):367–375.

Krivelevich, M., and Vilenchik, D. 2006. 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*, 454–463.

Lincoln, A., and Yedidia, A. 2019. Faster random k-cnf satisfiability. *CoRR* abs/1903.10618.

Paturi, R.; Pudlák, P.; Saks, M. E.; and Zane, F. 2005. An improved exponential-time algorithm for *k*-sat. *J. ACM* 52(3):337–364.

Paturi, R.; Pudlák, P.; and Zane, F. 1999. Satisfiability coding lemma. *Chicago J. Theor. Comput. Sci.* 1999.

Pudlák, P.; Scheder, D.; and Talebanfard, N. 2017. Tighter hard instances for PPSZ. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, 85:1–85:13.

Schöning, U. 1999. 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*, 410–414.

Selman, B.; Mitchell, D. G.; and Levesque, H. J. 1996. Generating hard satisfiability problems. *Artificial intelligence* 81(1-2):17–29.

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

Valiant, L. G., and Vazirani, V. V. 1986. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.* 47(3):85–93.

Vilenchik, D. 2019. personal communication.

Vyas, N., and Williams, R. R. 2019. On super strong ETH. In *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, 406–423.

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