~~Tight~~ Lower bounds implied by SETH

Connor Sawaske

Abstract

We examine implications of the Strong Exponential Time Hypothesis (SETH). After setting out to prove a statement bounding the runtimes of triangle detection algorithms (ultimately equivalent to the refutation of SETH), our failures to do so end up shedding some light on other implications.

1 Introduction

The notions of computational complexity and runtimes for algorithms have been of great interest to the field of computer science since its inception. Central to these studies are algorithms determining that satisfiability of a k-CNF formula (the k-SAT problem). The following conjecture (the Exponential Time Hypothesis) states that if $k > 2$ is fixed, then the runtime of any algorithm solving k-SAT must be exponential.

Conjecture 1.1 (ETH)**.** *For each $k > 2$, there exists $s_k > 0$ such that any algorithm solving k-SAT takes time at least $O(2^{s_k n})$.*

There is also a strengthening of ETH (called the Strong Exponential Time Hypothesis) stating that $O(2^n)$ is the best that we can do if k is not fixed.

Conjecture 1.2 (SETH)**.** *For every $\epsilon > 0$, there exists k such that k-SAT has no $O\left(2^{(1-\epsilon)n}\right)$-time algorithm.*

There is a fairly large preexisting body of work investigating the implications of ETH and SETH on the computational complexity of problems other than k-SAT. These conjectures can be used to provide lower bounds on runtimes for a variety of problems. In [ABHS17], the authors show that given $\epsilon > 0$, there exists $\delta > 0$ such that an algorithm finding a subset of n integers summing to a fixed integer T must run in time greater than $O(T^{1-\epsilon}2^{\delta n})$ unless SETH fails.

A very comprehensive survey of similar known results may be found in [LMS11]. For example, finding the chromatic number or a vertex cover or an independent set of a graph all have lower bounds on runtimes implies by SETH, and [CPP12] shows that there is a bound for finding an edge clique cover as well. Often, these lower bounds coincide with the runtimes of preexisting algorithms, exhibiting the tightness of the achieved bounds assuming ETH or SETH.

In this project, our initial aim was to contribute to this study of graph theory problems by answering the next question in the affirmative.

Question 1.3. *Fix $p > 2$, and suppose there exists an algorithm determining the presence of a p-cycle in a graph with E edges in time $\tilde{O}\left(E^{\frac{3}{2}-\delta}\right)$ for some $\delta > 0$. Does this imply that SETH fails?*

Of course, it turns out that there al-

ready exist algorithms with significantly lower runtimes than $\tilde{O}\left(E^{\frac{3}{2}-\delta}\right)$, so success would produce a proof of the refutation of SETH! Unsurprisingly, this was not ultimately achieved.

The rest of this report will focus on the methods used in probing this question and the results that came out of these efforts; it is structured as follows. In Section 2 we will introduce terminology and highlight some preexisting tools and results to be utilized. In Section 3, the main reduction translating satisfiability to cycle detection is constructed and analyzed. The limitations of this reduction are discussed in Section 4, and then used to show some implications of SETH. We close with some comments in Section 5.

2 Notation and tools

Our starting point is the following question.

Question 2.1. *Given a logical statement*

$$\varphi = \bigwedge_{i=1}^{m} (\ell_{i_1} \vee \ell_{i_2} \vee \cdots \vee \ell_{i_k})$$

where each ℓ_{i_j} is a literal of the form v_k or $\neg v_k$ for some set of variables $V = \{v_1, \ldots, v_n\}$, is there an assignment of true/false values to the elements of V such that φ is true?

In this setting, we say that φ is a **k-CNF** formula and we call the problem of determining the satisfiability of φ an instance of **k-SAT**. If we test each possible assignment of variables, then this problem can be answered in time $O(2^n)$.

We say that a variable $v \in V$ **supports** a clause γ of φ if either v or its negation appears in γ, and we call the set of variables that support γ the **supporting variables** of γ. We call the maximum number of clauses in which some element of V appears the **variable frequency** of φ.

Any arbitrary instance of k-SAT may be incredibly unwieldy. Since there are $2^k n^k$ possible clauses of length k in n variables, attempting to utilize any facts about the number of clauses or the variable frequency in a given instance is a difficult task. To circumvent this problem, a powerful reduction known as sparsification (originally due to Impagliazzo, Paturi, and Zane) is often invoked. We state here a strong version appearing in [CIP06].

Lemma 2.2 (Sparsification)**.** *Let φ be a k-CNF formula in n variables. For each $\epsilon > 0$, there exist k-CNF formulas $\psi_1, \ldots, \psi_t$ in the same n variables such that the following three conditions hold.*

(a) The disjunction $\bigvee_{i=1}^{t} \psi_i$ is satisfiable if and only if φ is satisfiable.

(b) The number of k-CNF formulas t satisfies $t \leq 2^{\epsilon n}$.

(c) Each variable occurs in at most $\left(\frac{k}{\epsilon}\right)^{3k}$ clauses of each ψ_i.

Furthermore, this reduction from φ to $\bigvee_{i=1}^{t} \psi_i$ runs in time $O(2^{\epsilon n})$.

This lemma is often invoked to show implications of SETH in the following way. Let $P(m)$ be some decision problem in the parameter m, where $P(m)$ can be solved in time $O(g(m))$ for some function $g(m)$. Suppose that the satisfiability of an instance of k-SAT in which each of n variable occurs in at most $\left(\frac{k}{\epsilon}\right)^{3k}$ clauses is equivalent to solving $P(f(n))$ for some function $f(n)$.

If $g(f(n))$ is in $O(2^{\delta n})$ for some $\delta < 1 - \epsilon$, then through Lemma 2.2 any instance of k-SAT can be solved in $O(2^{\epsilon n}2^{\delta n})$, violating SETH.

The central result demonstrating that our initial attempts to answer Question 1.3 would likely be in vain is the following (see [Wil16]).

Lemma 2.3. *There exists an algorithm determining the presence of a 3-cycle in a graph with v vertices that runs in time $O(v^{2.38})$.*

3 Cycles and satisfiability

Here we create a correspondence between cycles in a graph and necessary conditions for the satisfiability of an instance of k-SAT. To begin, let φ be a k-CNF formula with variable set $V = \{v_1, \ldots, v_n\}$. Denote by π a description of V as a union

$$V = V_1 \cup V_2 \cup V_3$$

of three subsets V_1, V_2, and V_3, each of which has size π_S. Note that π is not necessarily a strict partition of V; we allow for some overlap between the V_i's. We call π a **3-cover** of V. Given such a π, break φ up into a conjunction

$$\varphi = \varphi_\pi \wedge \varphi_\pi^c$$

where φ_π consists of all clauses in φ whose supporting variables occur in at most two out of the three sets V_1, V_2, and V_3 (so φ_π^c consists of all clauses whose supporting variables meet every one of V_1, V_2, and V_3). For instance, if

$$\varphi = (v_1 \vee v_2 \vee v_4) \wedge (v_3 \vee v_5 \vee v_6) \wedge (v_1 \vee v_4 \vee v_6)$$

and π is the 3-cover

$$\{v_1, \ldots, v_6\} = \{v_1, v_2\} \cup \{v_3, v_4\} \cup \{v_5, v_6\},$$

then

$$\varphi_\pi = (v_1 \vee v_2 \vee v_4) \wedge (v_3 \vee v_5 \vee v_6)$$

and

$$\varphi_\pi^c = (v_1 \vee v_4 \vee v_6).$$

Now we build a graph $G(\varphi, \pi)$ as follows. Let 2^{V_1}, 2^{V_2}, and 2^{V_3} be the sets of all possible true/false assignments to the variable sets V_1, V_2, and V_3, respectively. The vertex set of $G(\varphi, \pi)$ is defined to be the union $2^{V_1} \cup 2^{V_2} \cup 2^{V_3}$. Given a pair of vertices $\alpha_i \in 2^{V_i}$ and $\alpha_j \in 2^{V_j}$ with $i \neq j$, add the edge $\{\alpha_i, \alpha_j\}$ to $G(\varphi, \pi)$ if the following two conditions hold:

- If $V_i \cap V_j \neq \emptyset$, then α_i and α_j agree on this intersection.
- The common assignment of α_i and α_j to $V_i \cup V_j$ satisfies all clauses of φ_π whose supporting variables lie in $V_i \cup V_j$.

Having constructed $G(\varphi, \pi)$, we are now in a position to prove the next proposition, which is vital to the remainder of our results.

Proposition 3.1. *Let φ be a k-CNF formula on variable set $V = \{v_1, \ldots, v_n\}$ and let π be a 3-cover of V. Then there exists an algorithm determining the satisfiability of φ_π in time $O\left(2^{2.38\pi_S}\right)$*

Proof. From the construction of $G(\varphi, \pi)$, a cycle corresponds to a consistent assignment $(\alpha_1, \alpha_2, \alpha_3)$ of values to V such that every clause whose supporting variables meet at most two of V_1, V_2, and V_3 is satisfied. That is, a cycle in $G(\varphi, \pi)$ corresponds precisely to the satisfiability of the sub-formula φ_π.

By a brute-force approach to checking the existence of edges, the runtime for graph construction is in $O\left(2^{2\pi_S}\right)$. Since $G(\varphi, \pi)$ contains $3 \cdot 2^{\pi_S}$ vertices, Lemma 2.3 tells us that we can detect a cycle in time $O\left(2^{2.38\pi_S}\right)$. □

4 Results

There are three primary ways in which one might attempt to make use of Proposition 3.1 in answering Question 1.3. In each case, the failure of the approach demonstrates some implication of SETH on a separate algorithm. The first approach would be to construct a 3-cover π of V such that $\varphi = \varphi_\pi$. That is, make π such that the supporting variables of each clause of φ lie in some union $V_i \cup V_j$. While choosing π in this way makes for a nice satisfiability test, one pays the price of a larger runtime (by increasing π_S above $\frac{n}{3}$). Our first result details the restrictions that SETH imposes on such a process.

Theorem 4.1. *Let $V = \{v_1, \ldots, v_n\}$, and assume that SETH holds. Then:*

- *There is no algorithm producing for each k-CNF formula φ with variable set V a 3-cover π of V satisfying $\varphi = \varphi_\pi$ and $\pi_S = \frac{n(1-\delta_1)}{2.38}$ that runs in time $O\left(2^{(1-\delta_2)n}\right)$ for any $\delta_1, \delta_2 > 0$.*
- *If restricting only to k-CNF formulas with variable frequency at most $\left(\frac{k}{\epsilon}\right)^{3k}$, then there is no algorithm producing a 3-cover π of V satisfying $\varphi = \varphi_\pi$ and $\pi_S = \frac{n(1-\delta_1-\epsilon)}{2.38}$ that runs in time $O\left(2^{(1-\delta_2-\epsilon)n}\right)$ for any $\delta_1, \delta_2 > 0$.*

Remark 4.2. The first statement should not be too surprising. Indeed, for a general k-CNF formula, the number of clauses may be enormous and each variable can occur in some clause with every other single variable. In this case, a 3-cover may only be constructed with $\pi_S \geq \frac{n}{2}$. The instinct for producing a small 3-cover would be to first sparsify φ, and the second statement demonstrates that this does not improve the situation.

Proof. For the first statement, suppose to the contrary that there were such an algorithm for some $\delta_1, \delta_2 > 0$. Given a k-CNF formula φ with variable set V, let π be the 3-cover obtained from the algorithm. Then by Proposition 3.1, we have an algorithm determining the satisfiability of $\varphi_\pi = \varphi$ in time

$$O\left(2^{(1-\delta_2)n} + 2^{2.38\pi_S}\right) = O\left(2^{(1-\delta_2)n} + 2^{(1-\delta_1)n}\right),$$

contradicting SETH.

For the second statement, we proceed in the same manner. Given an arbitrary k-CNF formula φ with variable set V, first sparsify φ with respect to some $\epsilon > 0$ and obtain at most $2^{\epsilon n}$ k-CNF formulas with variable set V and variable frequency at most $\left(\frac{k}{\epsilon}\right)^{3k}$.

If an algorithm as in the second statement did exist, then we could run it on each of these new k-CNF formulas and apply Proposition 3.1 as before. Hence, we obtain an algorithm determining the satisfiability of φ with total runtime

$$\begin{aligned} &O\left(2^{\epsilon n}\left[2^{(1-\delta_2-\epsilon)n} + 2^{2.38\pi_S}\right]\right) \\ =&O\left(2^{\epsilon n}\left[2^{(1-\delta_2-\epsilon)n} + 2^{(1-\delta_2-\epsilon)n}\right]\right) \\ =&O\left(2^{\max\{(1-\delta_2)n,(1-\delta_1)n\}}\right), \end{aligned}$$

contradicting SETH. □

The second approach to using Proposition 3.1 forgoes the goal of choosing π so

that $\varphi = \varphi_\pi$. Instead, one may attempt to construct π so that the sub-formula φ_π^c is supported on some small subset of V. In particular, if the size of this set of supporting variables can be guaranteed to be some fraction of n, then a brute-force satisfiability test of φ_π^c may still yield a desirable runtime for the whole of φ. Once more, SETH has a strong implication on the feasibility of this approach.

Theorem 4.3. *Assuming SETH, there is no algorithm running in time $O\left(2^{(1-\delta_1)n}\right)$ producing a 3-cover π of the variable set $V = \{v_1, \ldots, v_n\}$ of a CNF formula φ such that $\pi_S = \frac{n}{3}$ and the variable set of φ_π^c has size $(1-\delta_2)n$, for any $\delta_1, \delta_2 > 0$.*

Proof. Suppose to the contrary that such an algorithm exists for some $\delta_1, \delta_2 > 0$. Given a k-CNF formula φ with variable set V, let π be the 3-cover obtained by the algorithm and let W be the subset of variables supporting $\varphi_\pi \pi^c$. First, collect all satisfying assignments of variables of φ_π^c by a brute-force method in time $O\left(2^{(1-\delta_2)n}\right)$, enumerate them, and store their restrictions to each of V_1, V_2, and V_3 along with the numbers of the assignments in which they appeared in a hash table.

Next, construct a subgraph of $G(\varphi, \pi)$ as follows. Instead of beginning with all possible assignments of values to V_1, V_2, and V_3 as vertices, use only to those whose restrictions to W appear as part of a satisfying assignment to $\varphi_\pi \pi^c$. When constructing edges, only add those whose common assignments appear in the same assignment in the hash table and annotate the edge with their image. At the last step of the triangle detection algorithm referenced in Lemma 2.3, one only needs to additionally verify that the annotations of the three edges intersect non-trivially. Considering the sizes involved in all of these steps, the total runtime becomes

$$\begin{aligned} &O\left(2^{(1-\delta_1)n} + 2^{(1-\delta_2)n} + 2^{2.48\pi_S}\right) \\ =&O\left(2^{\max\{(1-\delta_2)n, (1-\delta_1)n, \frac{2.48n}{3}\}}\right), \end{aligned}$$

contradicting SETH. □

Remark 4.4. In analogy with the second statement of Theorem 4.1, a similar result (omitted for brevity) holds that prohibits the use of sparsification in circumventing the roadblocks that may appear in a more general k-CNF formula.

The final way in which one might hope to make use of Proposition 3.1 is through modifying φ along with its variable set in such a way that a minimal 3-cover can be chosen and no "bad" clauses remain. Specifically, given a clause in φ of the form

$$\gamma = \gamma_1 \vee \gamma_2 \vee \gamma_3$$

in which each γ_i is a disjunction supported on V_i, we introduce a new variable v_γ, add it to some V_j, and replace this clause with the conjunction

$$\tilde{\gamma} = (\gamma_1 \vee \gamma_2 \vee v_\gamma) \wedge (\gamma_3 \vee (\neg v_\gamma)).$$

The satisfiability of φ is equivalent to the satisfiability of this modified version. Furthermore, each clause in $\tilde{\gamma}$ is supported on at most two out of the three variable sets. Let $\tilde{\varphi}$, $\tilde{V}$, and $\tilde{\pi}$ denote the new k-CNF formula, variable set, and 3-cover, respectively, obtained by applying this process to each bad clause in φ and distributing the new variables equally among V_1, V_2, and V_3.

Note that it is not strictly necessary to always introduce a new variable for each bad

clause. Consider, for example, that the satisfiability of

$$(x \vee y_1 \vee z_1) \wedge (x \vee y_2 \vee z_2)$$

is equivalent to the satisfiability of

$$(x \vee y_1 \vee a) \wedge (z_1 \vee (\neg a)) \wedge (x \vee y_2 \vee a) \wedge (z_2 \vee (\neg a)).$$

Nevertheless, the effectiveness of this modification process is limited by the next theorem. Though a more general statement also holds, we have focused this time on the "sparsified" version.

Theorem 4.5. *Assume that SETH holds. Then there is no algorithm producing, for each k-CNF formula φ with variable set $V = \{v_1, \ldots, v_n\}$ and 3-cover π with variable frequency at most $\left(\frac{k}{\epsilon}\right)^{3k}$ for some $\epsilon > 0$, the objects $\tilde{\varphi}$, $\tilde{V}$, and $\tilde{\pi}$ as above satisfying the following conditions.*

- $|\tilde{V}| = (1 + \delta_1)n$ *for some* $\delta_1 < \frac{.62 - 3\epsilon}{2.38}$.
- *The algorithm runs in time* $O\left(2^{(1-\delta_2-\epsilon)n}\right)$ *for some* $\delta_2 > 0$.

Proof. Suppose that such an algorithm existed for some $\delta_1, \delta_2 > 0$ and let φ be a general k-CNF formula on the variable set V with minimal 3-cover π. First sparsify φ with respect to some $\epsilon > 0$, and let ψ be one of the k-CNF formulas appearing in the sparsification. Now run the algorithm in the theorem on ψ. Since $\tilde{\psi}_{\tilde{\pi}} = \tilde{\psi}$ and $\tilde{\pi}_S = \frac{(1+\delta_1-\epsilon)n}{3}$, Proposition 3.1 tells us that we can determine the satisfiability of ψ in time

$$\begin{aligned} &O\left(2^{(1-\delta_2-\epsilon)n} + 2^{2.38\tilde{\pi}_S}\right) \\ =&O\left(2^{(1-\delta_2-\epsilon)n} + 2^{2.38(1+\delta_1)\frac{n}{3}}\right). \end{aligned}$$

Now repeat this for each formula appearing in the sparsification of φ. This achieves a total runtime for the satisfiability of φ in

$$O\left(2^{\epsilon n}\left(2^{(1-\delta_2-\epsilon)n} + 2^{2.38(1+\delta_1)\frac{n}{3}}\right)\right).$$

Since $\delta_1 < \frac{.62-3\epsilon}{2.38}$,

$$2.38(1 + \delta_1)\frac{n}{3} < (1 - \epsilon)n$$

and this contradicts SETH. □

5 Comments

In light of some recent discoveries, it should come as no surprise that each avenue outlined above attempting to produce a full reduction from satisfiability to cycle detection was met with significant obstacles. Indeed, in [GIKW] it is shown that cycle detection is part of a suite of problems which reduce to the k-orthogonal vectors problem. Since our method of splitting V into three parts and testing φ for satisfiability is essentially the 3-orthogonal vectors problem, success in a full reduction would show the equivalence of these two problems. In particular, this would imply that cycle detection is complete among first-order problems.

Acknowledgments

We thank Dan Suciu for suggesting the initial problem, along with providing an important reference in the latter stages of investigation. Additionally, the author would like to thank Isabella Novik and Ioana Dumitriu for enlightening conversations about hypergraph coloring.

References

[ABHS17] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Seth-based lower bounds for subset sum and bicriteria path. CoRR, abs/1704.04546, 2017.

[CIP06] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for sat. In Proceedings of the 21st Annual IEEE Conference on Computational Complexity, CCC '06, pages 252–260, Washington, DC, USA, 2006. IEEE Computer Society.

[CPP12] Marek Cygan, Marcin Pilipczuk, and Michal Pilipczuk. Known algorithms for EDGE CLIQUE COVER are probably optimal. CoRR, abs/1203.1754, 2012.

[GIKW] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for First-Order Properties on Sparse Structures with Algorithmic Applications, pages 2162–2181.

[LMS11] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. Bulletin of the EATCS, 105:41–72, 2011.

[Wil16] V. V. Williams. Cs267 lecture 1: Algorithms for fixed subgraph isomorphism, 2016.