Nondeterministic Extensions of the Strong Exponential Time Hypothesis and Consequences for Non-reducibility

Marco L. Carmosino Jiawei Gao Russell Impagliazzo
Ivan Mihajlin Ramamohan Paturi Stefan Schneider

{mcarmosi, jiawei, russell, imikhail, paturi, stschnei}@cs.ucsd.edu
Department of Computer Science and Engineering
UC, San Diego
La Jolla, CA 92093

ABSTRACT

We introduce the Nondeterministic Strong Exponential Time Hypothesis (NSETH) as a natural extension of the Strong Exponential Time Hypothesis (SETH). We show that both refuting and proving NSETH would have interesting consequences.

In particular we show that disproving NSETH would give new nontrivial circuit lower bounds. On the other hand, NSETH implies non-reducibility results, i.e. the absence of (deterministic) fine-grained reductions from SAT to a number of problems. As a consequence we conclude that unless this hypothesis fails, problems such as 3-SUM, APSP and model checking of a large class of first-order graph properties cannot be shown to be SETH-hard using deterministic or zero-error probabilistic reductions.

Categories and Subject Descriptors

F.1.3 [**Computation By Abstract Devices**]: Complexity Measures and Classes—*Reducibility and Completeness*; F.1.2 [**Computation By Abstract Devices**]: Modes of Computation—*Alternation and Nondeterminism*

Keywords

3-Sum; All-pairs shortest path; Computational Complexity; conditional lower bounds; fine-grained complexity; nondeterminism; SETH

1. INTRODUCTION

Traditionally, complexity theory has been used to distinguish very hard problems, such as NP-complete problems, from relatively easy problems, such as those in P. However, over the past few decades, there has been progress in understanding the exact complexities of problems, both for very hard problems and those within P, under plausible assumptions. For example, under hypotheses such as the 3-SUM conjecture [13] from computational geometry or the Strong Exponential Time Hypothesis for the complexity of SAT [16, 15], it follows that the known algorithms for many basic problems within P, including Fréchet distance [9], edit distance [5], string matching [1], k-dominating set [23], orthogonal vectors [26], stable marriage for low dimensional ordering functions [21], and many others [8], are essentially optimal.

Unfortunately, as our understanding of the relationship between the exact complexities of problems grows, so does the complexity of the web of known reductions and the number of distinct conjectures these results are based on. Ideally, we would like to show that many of these conjectures are in fact equivalent, or that all follow from some basic unifying hypothesis, thereby improving our understanding and simplifying the state of knowledge. For example, it would be nice to show that the 3-SUM conjecture follows from SETH (Strong Exponential Time Hypothesis). It would also be nice to show that SETH implies that HITTINGSET and MAXFLOW require superlinear time. Can we prove that APSP takes n^3 time under SETH?

In this paper, we introduce a new technique which provides evidence that such a simplification (i.e. hardness results under one unifying hypothesis such as SETH) is *unlikely*, at least when restricted to deterministic reductions. Just as one can show that a problem is unlikely to be NP-complete by showing that it belongs to a presumably smaller complexity class (such as $\mathsf{NP} \cap \mathsf{coNP}$), we can get *non-reducibility* results by comparing the complexity of problems in other models of computation.

To obtain our non-reducibility results, we consider the nondeterministic and co-nondeterministic complexities of the problem under question. If a problem has smaller nondeterministic and co-nondeterministic complexities, we show that if there were to be a deterministic *fine-grained* reduction from SAT to such a problem, it follows that SAT can be solved faster in co-nondeterministic time, which may be unlikely. More precisely, we introduce the following variant of SETH for nondeterministic models.

Nondeterministic Strong Exponential Time Hypothesis (NSETH): For every $\epsilon > 0$, there exists a k so that k-TAUT is not in $\mathsf{NTIME}[2^{n(1-\epsilon)}]$, where k-TAUT is the language of all k-DNF which are tautologies.

We feel that NSETH is plausible for many of the same reasons as SETH. Just as many algorithmic techniques have been developed for k-SAT, all of which approach exhaustive

ITCS'16, January 14–16, 2016, Cambridge, MA, USA.

DOI: http://dx.doi.org/10.1145/2840728.2840746.

search for large k, many proof systems have been considered for k-TAUT, and none have been shown to have significantly less than purely exponential complexity for large k. In fact, the tree-like ([24]) and regular resolution ([6]) proof systems have been proved to require such sizes. Moreover, we observe that results of [17] that obtain circuit lower bounds assuming SETH is false yield the same bounds assuming that NSETH is false. So disproving NSETH would be both a breakthrough in proof complexity and in circuit complexity.

We consider problems together with their *presumed* or *conjectured* complexities. Let the pair (L, T) denote the language L with (presumed) deterministic time complexity T. We use the notion of *fine-grained reducibility* (the special case of subcubic reducibility was defined in [33], the general case was defined in [32]) introduced by Vassilevska Williams [31] to reduce problems with their complexities to one another. We say that (L_1, T_1) is fine-grained reducible (denoted as $\leq_{\text{FGR}}$) to (L_2, T_2) if there is a Turing reduction from L_1 to L_2 such that improvement of the sort $T_2^{1-\epsilon}$ for $\epsilon > 0$ in the complexity of L_2 leads to an improvement of $T_1^{1-\delta}$ in the complexity of L_1 for some $\delta > 0$. We say that a language L with time complexity T is SETH-hard if there is a fine-grained reduction from CNFSAT with time 2^n to (L, T).

Using fine-grained reductions, an intricate web of relationships between improving basic algorithms within polynomial time has been established. By considering the nondeterministic and co-nondeterministic complexities of such problems, we show, under NSETH, that deterministic fine-grained reductions between many of these problems *do not exist*. In particular,

- HITTINGSET for sets of total size m and time $T(m) = m^{1+\gamma}$ is not SETH-hard for any $\gamma > 0$, and no problem that is SETH-hard reduces to HITTINGSET for any such time complexity.
- 3-SUM for $T(n) = n^{1.5+\gamma}$ is not SETH-hard for any $\gamma > 0$.
- MAXFLOW, minimum cost MAXFLOW, and maximum cardinality matching on a graph with m edges and $T(m) = m^{1+\gamma}$ are not SETH-hard.
- All-pairs shortest path on a graph with n vertices and $T(n) = n^{\frac{3+\omega}{2}+\gamma}$ is not SETH-hard.

While there are many known SETH-hard problems, few are graph problems, and those few have the same logical structure. In addition to specific problems, our method can be used to explain why the structure of SETH-hard graph problems are all similar. In particular, we consider first-order definable graph properties on sparse graphs (where we view the input size as the number of edges m). We show that, under SETH, the maximum time complexity for such a property expressible with k quantifiers will be close to $O(m^{k-1})$. On the other hand, if NSETH, all SETH-hard properties have the same logical structure: $k-1$ quantifiers of one type, followed by a single quantifier of the other type.

These results are only valid for deterministic or zero-error probabilistic fine-grained reductions. We introduce a non-uniform variant NUNSETH under which they also hold for randomized reductions with bounded error. However, some care should be used to evaluate whether this hypothesis is true, since it has not been the subject to previous study and Williams has recently shown related hypotheses about Merlin-Arthur complexity of k-TAUT are false ([30]).

2. OUTLINE OF THE PAPER

In section 3, we provide definitions of fine-grained reducibilities and establish basic closure properties of these reductions. In section 4, we outline reasons why disproving NSETH is nontrivial. In section 5, we examine the nondeterministic and co-nondeterministic complexities of several problems within polynomial time whose exact complexities have been extensively studied, and show that, under NSETH, none of these problems are SETH-hard. In section 6, we explain why all the known maximally hard SETH-hard first-order graph properties have the same logical structure.

In section 7, we show that NSETH also implies that certain new problems are hard, especially those involving verifying solutions to known SETH-hard problems. Finally, section 8 presents our conclusions and open problems.

3. DEFINITIONS AND BASIC PROPERTIES

Fine-grained reductions are defined with the motivation to control the exact complexity of the reducibility. For this purpose, we consider languages together with their *presumed* or *conjectured* complexities. We use the pair (L, T) to denote a language together with its time complexity T. Intuitively, if (L_1, T_1) fine-grained reduces to (L_2, T_2), then any constant savings in the exponent of the time complexity of L_2 implies some constant savings in the exponent of the time complexity of L_1.

Definition 1 (Fine-Grained Reductions ($\leq_{FGR}$))**.** *Let L_1 and L_2 be languages, and let T_1 and T_2 be time bounds. We say that (L_1, T_1)* fine-grained reduces *to (L_2, T_2) (denoted $(L_1, T_1) \leq_{FGR} (L_2, T_2)$) if*

(a) $\forall \epsilon > 0 \quad \exists \delta > 0$, $\exists \mathcal{M}^{L_2}$, a deterministic Turing reduction from L_1 to L_2, such that

$$\mathsf{TIME}[\mathcal{M}] \leq T_1^{1-\delta}$$

(b) Let $\tilde{Q}(\mathcal{M}, x)$ denote the set of queries made by $\mathcal{M}$ to the oracle on an input x of length n. The query lengths obey the following time bound.

$$\sum_{q \in \tilde{Q}(\mathcal{M},x)} (T_2(|q|))^{1-\epsilon} \leq (T_1(n))^{1-\delta}$$

If a fine-grained reduction exists from (L_1, T_1) to (L_2, T_2), algorithmic savings for L_2 can be transferred to L_1. The definition gives us exactly what is needed to establish savings for L_1 by simulating the machine $\mathcal{M}^{L_2}$ using the faster algorithm for L_2. The role of each parameter in the definition of fine-grained reducibility makes this clear.

T_1**:** The presumed time to decide L_1, usually given by a trivial algorithm.

T_2**:** The presumed time to decide L_2.

ϵ**:** Any savings (assumed or real) on computing L_2.

δ: The savings (as a function of ϵ) that can be obtained over T_1 when deciding L_1 by reducing to L_2.

Definition 2 (Randomized Fine-Grained Reductions ($\leq^s_{rFGR}$)). *Exactly as in the deterministic case, except the Turing reduction from (L_1, T_1) to (L_2, T_2) is a probabilistic machine with some two-sided error bound*

$$\Pr[\mathcal{M}^{L_2}(x) = L_1(x)] \geq s$$

We denote a randomized fine grained reduction from L_1 to L_2 with error bound s by $(L_1, T_1) \leq^s_{rFGR} (L_2, T_2)$. Generally, we will use $s = 2/3$, so we denote $\leq^{2/3}_{rFGR}$ by $\leq_{rFGR}$.

We will have occasion to consider FGRs between function problems. This poses the problem that, in certain situations, just writing down the solution to a problem could exceed the time bound and wipe out fine-grained savings. In the deterministic case, we cope with this by adding another restriction to the definition of a fine-grained reduction:

Definition 3 (Fine-Grained Reductions for Functions ($\leq_{fFGR}$)). *Exactly as in the decision deterministic case, except that the Turing reduction $\mathcal{M}^{f_2}$ is to a function problem f_2 and is expected to produce a functional output. In addition to the existing resource bounds, we bound the size of answers given by the f_2 oracle.*

$$\sum_{q \in \tilde{Q}(\mathcal{M},x)} (|f_2(q)|) \leq (T_1(n))^{1-\delta}$$

The bound on query answer size ensures that each proof about decision FGRs goes through in the function FGR case, with an additional step corrosponding to the bound on query answers that is identical to checking the bound on query sizes of the definition of a decision FGR.

We will also consider FGRs between nondeterministic computation of function problems. Defining exactly what it means for a nondeterministic machine to compute a function is fairly involved, so we sidestep this issue by using the *graph* of the function as a decision problem. That is, by convention we use the language $gr(f) = \{\langle x, f(x)\rangle | x \in \{0,1\}^*\}$ to assess the nondeterministic complexity of every function f we are interested in. Since here we only only study $(N \cap coN)TIME$ complexity, this convention does not unduly simplify our model. It is equivalent to being able to print the ith bit of $f(x)$ on input x in $(N \cap coN)TIME$, which we would have anyway. Thus, using the graph of a function, all properties of FGRs between the nondeterministic complexity of decision problems hold between function problems as well.

3.1 Deterministic Fine-grained Reductions

The properties of deterministic fine-grained reductions are exactly what one would expect and follow by standard methods. See full version of the paper for proofs.

Lemma 1 (Fine-grained reductions translate savings for DTIME). *Let $(L_1, T_1) \leq_{FGR} (L_2, T_2)$, and $L_2 \in \mathsf{DTIME}[T_2(n)^{1-\epsilon}]$ for $\epsilon > 0$. There exists $\delta > 0$ such that*

$$L_1 \in \mathsf{DTIME}[T_1(n)^{1-\delta}]$$

Lemma 2 (Fine-grained reductions transfer savings for $(N \cap coN)TIME$). *Let $(L_1, T_1) \leq_{FGR} (L_2, T_2)$, and $L_2 \in (N \cap coN)TIME[T_2(n)^{1-\epsilon}]$ for some $\epsilon > 0$. Then there exists a $\delta > 0$ such that*

$$L_1 \in (N \cap coN)TIME[T_1(n)^{1-\delta}]$$

To prove both of these "savings" lemmas, we simply run the reduction TM and simulate oracle calls to L_2 using the efficient algorithm for L_2 to get savings for L_1.

Corollary 1 (Fine-grained reductions translate savings from reductions). *When the true complexity of a problem is meaningfully smaller than the time bound used in a fine-grained reduction, savings are translated.*

1. *Let $(L_1, T_1) \leq_{FGR} (L_2, T_2^{1+\gamma})$, and $L_2 \in \mathsf{DTIME}[T_2]$. Then there exists $\delta > 0$ such that*
$$L_1 \in \mathsf{DTIME}[T_1^{1-\delta}]$$
2. *Let $(L_1, T_1) \leq_{FGR} (L_2, T_2^{1+\gamma})$, and $L_2 \in (N \cap coN)TIME[T_2]$. Then there exists a $\delta > 0$ such that*
$$L_2 \in (N \cap coN)TIME[T_1^{1-\delta}]$$

The above follows from the saving transfer lemmas by a simple substitution.

Lemma 3 (Fine-grained reductions are closed under composition). *Let $(A, T_A) \leq_{FGR} (B, T_B)$ and $(B, T_B) \leq_{FGR} (C, T_C)$. It then follows $(A, T_A) \leq_{FGR} (C, T_C)$.*

Finally, composition is proved by carefully verifying time and query bounds on the obvious "nested" simulation of A using the algorithm for C.

3.2 Randomized FGRs

As we will show, many of the problems such as k-SUM and HITTINGSET which have served as starting points for fine-grained reductions have substantially smaller nondeterministic complexities than their conjectured deterministic complexities. From the above closure properties, it will follow that if NSETH is true, none of these problems is SETH-hard under deterministic (or zero-error probabilistic) fine-grained reductions. This leaves a major loophole: these problems might still be SETH-hard under randomized reductions. In this section, we will outline a reason why even randomized SETH-hardness would be somewhat surprising. We introduce a non-uniform version of NSETH, NUNSETH, and show that this hypothesis would imply the non-existence of even randomized SETH-hardness results.

Definition 4. *Let k-TAUT be the tautology problem restricted to k-DNF's. The* Non-uniform Nondeterministic Strong Exponential Time Hypothesis *(*NUNSETH*) is the statement : $\forall \epsilon > 0\, \exists k \geq 0$, so that there are no nondeterministic circuit families of size $O(2^{n(1-\epsilon)})$ recognizing the language k-TAUT.*

While we do not have any general conservation of nonuniform nondeterministic time by randomized reductions, we do have a limit for the special case of problems that are SETH-hard under randomized reductions.

Lemma 4. *Assume L is* SETH*-hard with $T(N)$ via a randomized reduction. If* NUNSETH*, then there is no $\delta > 0$ so that $L \in (N \cap coN)TIME[T^{1-\delta}(n)]$.*

Proof. Let ϵ be the constant corresponding to δ in the reduction, and let $\mathcal{M}^L$ be the corresponding randomized oracle machine. Let $m < n^k$ be the length in bits of a description of a k-SAT formula on n inputs. By repeating $\mathcal{M}^L$ $O(m)$ times and taking the majority answer, we can make the error probability less than 2^{-m}. Therefore, there is one random tape that has no errors, using the standard argument that $\mathsf{BPP} \in \mathsf{P/poly}$. Since $\mathcal{M}$ runs in total time $2^{(1-\epsilon)n}$, this tape will have length at most $m2^{(1-\epsilon)n}$, and so will be an exponential improvement over 2^n. Once we have fixed the tape, we can simulate the oracle queries nondeterministically as in the case of deterministic reductions, with total complexity $O(m)$ times what it is for one run. Thus, we get a nondeterministic circuit with total size $O(m2^{(1-\epsilon)n})$. □

Note that the above argument, in addition to needing advice, multiplies the complexity by an amount polynomial in the input size. While this is not an issue for SAT, it would render the consequences of randomized reductions for problems within P moot, since we are trying to preserve exact polynomial complexities.

While NUNSETH seems plausible, we should exercise some caution before adopting it as an axiom. First, there are no known consequences if NUNSETH fails to be true. Secondly, we originally were going to add equally plausible (to us) hypotheses concerning the total time for bounded round interactive protocols for k-TAUT. However, Williams recently showed that even the general formula counting problem has a Merlin-Arthur protocol of total complexity $\tilde{O}(2^{n/2})$. Because there is a polynomial overhead in making such a protocol a nondeterministic algorithm with advice, this does not contradict NUNSETH. However, it does remind us that counter-intuitive things can happen when randomness and nondeterminism are combined, so we should be cautious in assuming non-uniformity might not speed up computation in this circumstance.

4. WHAT IF ¬NSETH?

SETH is an interesting hypothesis because both ¬SETH and SETH have interesting consequences that seem difficult to prove unconditionally. In this section, we show that the same proofs that show "¬SETH implies circuit lower bounds" can be applied to ¬NSETH as well. This is evidence that NSETH will be hard to refute.

Algorithms for CKT-SAT or CKT-TAUT imply circuit lower bounds (see [27] and [29]). For some restricted circuit classes $\mathcal{C}$, we can reduce satisfiability or tautology of $\mathcal{C}$-circuits to k-SAT or k-TAUT by decomposing $\mathcal{C}$ circuits into a "big OR" of CNF formulas. Thus, both ¬SETH and ¬NSETH imply faster $\mathcal{C}$-circuit analysis algorithms (tautology or satisfiability) for these classes, which imply lower bounds.

The proofs of [17] optimize the reduction of arbitrary nondeterministic time languages to 3-SAT to obtain new "failure of a hardness hypothesis about k-SAT implies circuit lower bounds" results for a variety of circuit classes. The following (see the full version for details) is implicit in their work:

Theorem 1. *We have the following implications from failure of a k-TAUT hardness hypothesis to circuit lower bounds for restricted classes:*

1. *If the nondeterministic exponential time hypothesis (*NETH*) is false; i.e., for every $\epsilon > 0$, 3-TAUT is in time $2^{\epsilon n}$, then $\exists f \in \mathsf{E}^{\mathsf{NP}}$ such that f does not have linear-size circuits.*

2. *If the nondeterministic strong exponential time hypothesis (*NSETH*) is false; i.e., there is a $\delta < 1$ such that for every k, k-TAUT is in time $2^{\delta n}$, then $\exists f \in \mathsf{E}^{\mathsf{NP}}$ such that f does not have linear-size series-parallel circuits.*

3. *If there is $\alpha > 0$ such that n^α-TAUT is in time $2^{n-\omega(n/\log\log n)}$, then $\exists f \in \mathsf{E}^{\mathsf{NP}}$ such that f does not have linear-size log-depth circuits.*

Since (by item 2 above) refuting NSETH would give nontrivial circuit lower bounds, it is unlikely to be easy to refute.

5. THE NONDETERMINISTIC TIME COMPLEXITY OF PROBLEMS IN P

How could we show that one language is not reducible to another language? There is an ever-growing web of problems, hypotheses, and reductions that reflect the fine-grained complexity approach to explaining hardness. Could this structure collapse into a radically simpler graph, with just a few equivalence classes? If we assume NSETH, the answer to this question is *probably not as much as one might hope.*

We can broadly categorize computational problems into two sets. In the first category, the deterministic time complexity is higher than both the nondeterministic and co-nondeterministic time complexity. In the second category, at least one of nondeterminism or co-nondeterminism does not help in solving the problem more efficiently. Corollary 1 shows that savings in $(N \cap coN)TIME$ are preserved under deterministic fine-grained reductions. As a result, we can rule out tight reductions from a problem that is hard using nondeterminism or co-nondeterminism to a problem that is easy in $(N \cap coN)TIME$.

If NSETH holds, then k-TAUT is in the category of problems that do not benefit fron nondeterminism.. benefit from co-nondeterminism. So, any problem that is SETH-hard under deterministic reductions also falls in this category.

In this section we explore problems that do benefit from $(N \cap coN)TIME$, i.e. we give nondeterministic algorithms that are faster than their presumed deterministic time complexities. This rules out deterministic fine-grained reductions from CNFSAT to these problems with their presumed time complexities. As a consequence, it is not possible to show that these problems are SETH-hard using a deterministic reduction.

We begin by formalizing the notion of non-reducibility.

Theorem 2 (NSETH implies no reduction from SAT)**.** *If* NSETH *and $C \in (N \cap coN)TIME[T_C]$ for some problem C, then $(\mathsf{SAT}, 2^n) \not\leq_{FGR} (C, T_C^{1+\gamma})$ for any $\gamma > 0$.*

Proof. Assume NSETH, $(\mathsf{SAT}, 2^n) \leq_{FGR} (C, T_C^{1+\gamma})$, and $C \in (N \cap coN)TIME[T_C]$. By Corollary 1, preservation of $(N \cap coN)TIME$ savings under fine-grained reductions, there exists $\delta > 0$ such that $\mathsf{SAT} \in (N \cap coN)TIME[2^{n(1-\delta)}]$. This contradicts NSETH, therefore it cannot be the case (under NSETH) that $(\mathsf{SAT}, 2^n) \leq_{FGR} (C, T_C)$. □

Corollary 2 (NSETH implies no reductions from SETH-hard problems)**.** *If* NSETH *holds and $C \in (N \cap$*

coN)TIME$[T_C]$*, then for any* B *that is* SETH*-hard under deterministic reductions with time* T_B*, and* $\gamma > 0$*, we have*

$$(B, T_B) \not\leq_{FGR} (C, T_C^{1+\gamma})$$

Proof. Assume NSETH, and that (B, T_B) is SETH-hard. Therefore, we know $(\mathsf{SAT}, 2^n) \leq_{FGR} (B, T_B)$. Now assume $(B, T_B) \leq_{FGR} (C, T_C^{1+\gamma})$. Then by Lemma 3, composition of fine-grained reductions, we have that $(\mathsf{SAT}, 2^n) \leq_{FGR} (C, T_C)$. But by Theorem 2 above, this is impossible under NSETH. □

We now give the main result of this section.

Theorem 3. *Under* NSETH*, there is no deterministic or zero-error fine-grained reduction from* SAT *or any* SETH*-hard problem to the following problems with the following time complexities for any* $\gamma > 0$*.*

- MAXFLOW*, min-cost* MAXFLOW*, and maximum matching with* $T(m) = m^{1+\gamma}$
- HITTINGSET *with* $T(m) = m^{1+\gamma}$
- 3-SUM *with* $T(n) = n^{1.5+\gamma}$
- *All-pairs shortest path with* $T(n) = n^{\frac{3+\omega}{2}+\gamma}$

Note that for graph problems, n refers to the number of vertices, m refers to the number of edges, and ω is the matrix multiplication exponent.

To prove Theorem 3 we give both nondeterministic and co-nondeterministic algorithms for these problems.

5.1 Maximum Flow

The maximum flow problem has been an extensively studied problem for decades and has a large number of theoretical and practical applications. While approximate maximum flow on undirected graphs has a $\tilde{O}(m)$ algorithm [18], where m is the number of edges, no linear time algorithm is known for the exact version of the problem.

A natural question from the point of conditional hardness is if we can prove a superlinear lower bound by proving that the problem is SETH-hard.

In this section we use the max-flow/min-cut theorem to give a $(N \cap coN)TIME$ algorithm for the decision version of max-flow with time linear in the number of edges. Assuming NSETH, we can then conclude that there is no deterministic fine-grained reduction from any SETH-hard problem to maximum flow with a superlinear time bound.

Definition 5 (Maximum Flow Problem)**.** *Let* $G = (V, E)$ *be a connected directed graph,* $s, t \in V$ *be vertices and* $k \in \mathbb{R}$*.*

*The maximum flow problem (*MAXFLOW*) is to decide if there exists a flow from* s *to* t *of value at least* k*.*

The nondeterministic algorithm for maximum flow is straightforward and the co-nondeterministic algorithm follows directly for the max-flow/min-cut theorem.

Lemma 5. MAXFLOW $\in (N \cap coN)TIME[O(m)]$

Proof. For the nondeterministic algorithm, nondeterministically guess the flow on each edge. We can verify in linear time that the value of the flow is at least k, that no edge flow exceeds the edge capacity, and that for all nodes the inflow is equal to the outflow.

For the co-nondeterministic algorithm, nondeterministically guess a cut (S, T) such that $s \in S$ and $t \in T$ with value l where $l < k$. By the max-flow/min-cut theorem there is no flow with value strictly greater than l. The value of a cut can be computed in $O(m)$ time. □

This completes the part of Theorem 3 concerning maximum flow. In contrast, the single-source maximum flow problem requires quadratic time under SETH [2]. In the single-source maximum flow problem we are given a source s and need to output the maximum flow from s to all other nodes. As a consequence, there is no deterministic fine-grained reduction from single-source maximum flow to maximum flow under NSETH.

5.2 Hitting Set

Given two families of non-empty sets $\mathcal{S}$ and $\mathcal{T}$ defined on universe U, a set $S \in \mathcal{S}$ is a *hitting set* if it has nonempty intersections with all members in $\mathcal{T}$. The HITTINGSET problem accepts input $(\mathcal{S}, \mathcal{T}, U)$ iff

$$\exists S \in \mathcal{S}\ \forall T \in \mathcal{T}\ \exists u \in U\ \ ((u \in S) \wedge (u \in T))$$

Let the size of input be $m = \sum_{S \in \mathcal{S}} |S| + \sum_{T \in \mathcal{T}} |T|$. We assume for any $u \in U$, we can in constant time decide if $u \in S$ or $u \in T$. It is conjectured, that this problem does not admit a subquadratic time algorithm [4]. We show that HITTINGSET and its negation are both solvable in nondeterministic linear time.

Lemma 6. HITTINGSET $\in (N \cap coN)TIME[O(m)]$

HITTINGSET can be solved nondeterministically in linear time, by guessing an S, enumerating all $T \in \mathcal{T}$, and guessing a $u \in T$.

The negation of the HITTINGSET problem ¬HITTINGSET, which is defined as

$$\forall S \in \mathcal{S}\ \exists T \in \mathcal{T}\ \forall u \in U\ \ ((u \notin S) \vee (u \notin T))$$

can be solved by the following algorithm.

```
for each S ∈ 𝒮 do
    Nondeterministically select T from 𝒯;
    for each u ∈ S do
        if u ∈ T then
            Reject.
        end
    end
end
Accept.
```

Algorithm 1: Algorithm for ¬HITTINGSET

The algorithm runs in time $O(\sum_{S \in \mathcal{S}} |S|) = O(m)$.

The full version generalizes this algorithm for model checking of arbitrary k-quantifier sentences with at least one existential quantifier and ending with a universal quantifier.

5.3 Min-Cost Maximum Flow

The min-cost maximum flow problem is an important generalization of the maximum flow problem that also generalizes problems such as shortest path and bipartite minimum cost perfect matching.

In the min-cost maximum flow problem on a graph $G = (V, E)$ we consider flow networks where the edges e have aditional costs $\psi(e)$. The cost of a flow is defined as

$$\sum_{e \in E} \psi(e)\text{flow}(e)$$

Definition 6. *Let $G = (V, E)$ be a connected directed graph with capacity constraints and edge costs, let $s, t \in V$ be vertices and $k, c \in \mathbb{R}$.*

The min-cost maximum flow problem *is to decide if there either exists a flow from s to t of value strictly more than k, or if there is a flow from s to t of value exactly k and cost at most c.*

Orlin [22] gives a $O(m^2)$ algorithm for min-cost maxflow. In this section consider the question if it is possible to show SETH-hardness of this problem and show that there is a $O(m)$ nondeterministic and co-nondeterministic algorithm. Therefore, assuming NSETH, this problem is not SETH-hard under deterministic reductions for any superlinear time.

It is easy to see that this problem is in $\mathsf{NTIME}[O(m)]$ where m is the number of edges. Simply either guess a maximum flow with minimum cost and verify that it is indeed a flow with the correct value and cost. We therefore concentrate on the co-nondeterministic time complexity.

Lemma 7. *The min-cost maximum flow problem is in $(N \cap coN)TIME[O(m)]$.*

Proof. Klein [19] showed that a for any flow f, there is a flow of the same value as f but smaller cost if and only if there is a negative cost cycle in the residual graph.

Furthermore, as oserved in the analysis of the Bellmann-Ford algorithm [7, 12], there is a nondeterministic algorithm for the nonexistance of a negative weight cycle in a graph. A potential for a weighted graph $G = (V, E, w)$ is a map $p : V \to \mathbb{R}$ such that for all edges $(u, v) \in E$ we have $p(v) \leq p(u) + w(u, v)$. Bellman and Ford show that there is a negative weight cycle in G if and only if there is no potential for G.

The co-nondeterministic algorithm for min-cost maximum flow has two cases. If there is no flow of value k, then we nondeterministically guess a cut of value less than k. Otherwise, nondeterministically guess a flow of value k with minimum cost. We then certify that the flow is a maximum flow by guessing a cut of value k. Furthermore we guess a potential for the residual graph. The cut certifies that there is no flow of value greater than k, and the potential certifies that there is no maximum flow of smaller cost.

Verifying all nondeterministic guesses can be done in time $O(m)$. □

Since the maximum flow problem is a generalization of the min-cost maximum flow problem, Lemma 5 also follows as a corollary of 7.

5.4 Maximum Matching

The maximum matching problem in general graphs is one of the most fundamental problems in computer science. The maximum matching problem is in time $O(m\sqrt{n})$ [20], matching the time complexity of the bipartite case [14].

In this section we show that there is a linear time co-nondeterministic algorithm, and that there is therefore no fine-grained reduction from CNFSAT to maximum matching for any superlinear time, assuming NSETH.

Definition 7 (Maximum Matching Problem)**.** *The maximum matching problem is given a graph $G = (V, E)$ and a number k, is to decide if there exists a matching of size at least k.*

We give an $O(m)$ co-nondeterministc algorithm for this problem. The $O(m)$ nondeterministic algorithm is trivial.

Lemma 8. *The maximum matching problem is in $(N \cap coN)TIME[O(m)]$.*

Proof. Edmonds Theorem [11] relates maximum matchings of a graph $G = (V, E)$ with odd set covers. An odd set cover is a map $f : V \to \mathbb{N}$, such that each edge is either adjacent to a vertex v with $f(v) = 1$, or is adjacent to two vertices u, v such that $f(u) = f(v) \geq 2$. Furthermore, for $n_i = |\{v \mid v \in V, f(v) = i\}|$ we have n_i is odd for all $i \geq 2$.

For an odd set cover O, let $\text{val}(O) = n_1 + \sum_{i \geq 2} \lfloor \frac{n_i}{2} \rfloor$ be the value of the set cover. Edmonds Theorem says that for any matching M and any odd set cover O, we have $|M| \leq \text{val}(O)$. Furthermure, for any maximum matching M there is an odd set cover O such that $|M| = \text{val}(O)$. Therefore a matching M is maximum if and only if there is an odd set cover O such that $|M| = \text{val}(O)$.

The co-nondeterministic algorithm then guesses a maximum matching M and an odd set cover O such that $|M| = \text{val}(O)$.

Verifying that M is a matching and O an odd set cover, as well as computing the value of the set cover can easily be done in time $O(m)$. □

5.5 3-SUM

The conjecture that the 3-SUM problem admits no $O(n^{2-\epsilon})$ algorithm for any $\epsilon > 0$ has proven immensely useful to show the conditional hardness of a large number of problems (e.g. [13, 10, 3]), most of which are not known to be hard under SETH. A fine-grained reduction from SAT to 3-SUM would therefore have a large impact, proving the 3-SUM conjecture under SETH.

We give a subquadratic algorithm for 3-SUM in $(N \cap coN)TIME$, which rules out a deterministic fine-grained reduction from SAT to 3-SUM under NSETH.

Definition 8. *Given n integers $a_1 \dots a_n$ in the range $[-n^c, n^c]$ for some constant c, the* 3-Sum problem *(3-SUM) is the problem of determining if there is a triple $1 \leq i, j, k \leq n$ such that $a_i + a_j + a_k = 0$.*

Lemma 9. 3-SUM $\in (N \cap coN)TIME[\tilde{O}(n^{1.5})]$

Proof. There is a trivial constant time nondeterministic algorithm of guessing the triplet of indices. The more interesting part is to show that there is an efficient nondeterministic algorithm to show that there is no such triplet.

We nondeterministically guess a proof of the form (p, t, S), such that

- p is a prime number, such that $p \leq \mathbf{prime}_{n^{1.5}}$, where $\mathbf{prime}_i$ is i-th prime number.
- t is a nonnegative integer with $t \leq 3cn^{1.5} \log n$ such that $t = |\{(i, j, k) \mid a_i + a_j + a_k = 0 \mod p\}|$ is the number of three-sums modulo p.
- $S = \{(i_1, j_1, k_1), \dots, (i_t, j_t, k_t)\}$ is a set of t triples of indices, such that for all $r : 0 < r \leq t$ we have $a_{i_r} + a_{j_r} + a_{k_r} = 0 \mod p$ and $a_{i_r} + a_{j_r} + a_{k_r} \neq 0$

We first show that such a proof exists. Let us assume that there is no triple of elements that sum up to zero. Let R be the set of all pairs $((i,j,k),p)$, such that p is a prime $\leq \mathbf{prime}_{n^{1.5}}$ and $a_i + a_j + a_k = 0 \mod p$. Then $|R| \leq n^3 \log(3n^c) < 3cn^3 \log n$, as any integer z can have at most $\log(z)$ prime divisors. Then, by a simple counting argument, there indeed exists a prime $p_0 \leq \mathbf{prime}_{n^{1.5}}$, such that the number of pairs of the form $((i,j,k),p_0)$ in R is at most $\frac{3cn^3 \log n}{n^{1.5}} = 3cn^{1.5} \log n$).

To verify a proof of that form we first need to check that for all $r \leq t$:

$$a_{i_r} + a_{j_r} + a_{k_r} = 0 \mod p$$

$$a_{i_r} + a_{j_r} + a_{k_r} \neq 0$$

Then we compute the number of 3-sums modulo p and compare it with t. In order to do this we expand the following expression using Fast Fourier Transform in time $\tilde{O}(t)$:

$$\left(\sum_i x^{(a_i \mod p)}\right)^3$$

Let b_j be a coefficient before x^j. We need to check that

$$b_0 + b_p + b_{2p} = t$$

If it is true, then the proof is accepted, otherwise it is rejected.

The time complexity of verification is $\tilde{O}(n^{1.5})$ for reading and checking the properties of all the triples and $\tilde{O}(t) = \tilde{O}(n^{1.5})$ for counting the number of triples that sum to 0 modulo p. Therefore the total time complexity is $\tilde{O}(n^{1.5})$. □

5.6 All-pairs shortest paths and related problems

The All-pairs shortest path problem (APSP) is to find the shortest path in a graph between any pair of nodes. Like the 3-SUM conjecture and SETH, the conjecture that APSP does not admit an $O(n^{3-\epsilon})$ time algorithm for any $\epsilon > 0$ has been used successfully to show the conditional hardness of a number of problems, e.g. [33, 25].

We use a similar technique as in the algorithm for 3-SUM to show that the Zero Weight Triangle problem (ZWT), which is hard under APSP, admits an efficient algorithm in $(N \cap coN)TIME$.

Definition 9. *Given a tripartite graph $G(V_1, V_2, V_3, E)$ with $|V_1| = |V_2| = |V_3| = n$ and edge weights in $[-n^a, n^a]$, the* Zero Weight Triangle problem *is the problem of determining if there is a triangle such that the sum of the edge weights is* 0.

We first show that if the range is small enough, then we can count the number of zero weight triangles efficiently.

Lemma 10. *For a prime p, there is a deterministic algorithm for counting the number of zero weight triangles* mod p *in time $O(n^\omega p)$*

Proof. For $i \in \mathrm{GF}(p)$, let $q(i)$ be the polynomial x^i. Let A be the weight matrix of the input graph G (mod p). We define matrix B as $B[i,j] = q(A[i,j])$. For a polynomial r and integer i, let $b_{i,r}$ be the coefficient of x^i in r. Every triangle with weight zero mod p has weight either 0, p and $3p$. We have that $b_{j,B^3[i,i]}$ is the number of traingles of weight j that involve vertex i. Therefore

$$\sum_{i=1}^{n} \sum_{j \in \{0,p,2p\}} b_{j,B^3[i,i]} = 3t \tag{1}$$

where t is the number of zero weight triangles modulo p.

The time to compute B^3 is $O(n^\omega p)$ if we multiply the polynomials using Fast Fourier Transform. □

In particular, we will be using Lemma 10 to verify that our nondeterministic guess of the number of false positives is correct.

Lemma 11. *The Zero Weight Triangle Problem is in $(N \cap coN)TIME[O(n^{\frac{\omega+3}{2}})]$.*

Proof. As for 3-SUM, the nondeterministic algorithm is trivial and we concentrate on the co-nondeterministic algorithm.

Let $\mu = \frac{3-\omega}{2}$. Further let c be a large constant such that there are at least n^μ primes in the range $R = [n^\mu, cn^\mu \log n]$. We assume that there is no zero weight triangle and consider any fixed triangle. The total weight of the triangle is in the range $[-3n^a, 3n^a]$ and the number of primes $p \in R$ such that the triangle has weight 0 mod p is at most $\log(3n^a)/\log(n^\mu) < \frac{2}{\mu}a$. Since R contains at least n^μ primes, there is a prime $p \in R$ such that the number of triangles with weight 0 mod p is at most $\frac{2}{\mu}an^{\frac{3+\omega}{2}}$.

The nondeterministic algorithm now proceeds as follows: Nondeterministically pick p as above. By Lemma 10 we can deterministically count the number t of triangles with weight 0 mod p in time $O(n^\omega p) = O(n^{\frac{3+\omega}{2}})$. Nondeterministically pick t distinct triangles and check that each of them has weight $w \neq 0$ with $w = 0 \mod p$.

The total time is bounded by $O(n^{\frac{3+\omega}{2}})$ as claimed. □

Corollary 3. $\mathrm{APSP} \in (N \cap coN)TIME[\tilde{O}(n^{\frac{3+\omega}{2}})]$.

Proof. A deterministic fine-grained reduction from the problem of finding a negative weight triangle to ZWT can be found in [25], such that the negative weight triangle problem is also in $(N \cap coN)TIME[\tilde{O}(n^{\frac{3+\omega}{2}})]$.

Finally, [33] give a deterministic fine-grained reduction from APSP to the negative weight triangle problem with time $\tilde{O}(n^2 T(n^{1/3}))$, where $T(n)$ is the time complexity of the negative weight triangle problem.

Instead of applying this reduction directly, which would still give a subcubic nondeterministic upper bound for APSP, we instead modify their reduction to a nondeterministic reduction that preserves the savings in the exponent. The reduction from [33] loses savings in the exponent when reducing from min-plus product to negative weight triangle. The fine-grained reduction from APSP to min-plus product is folklore and does not change the exponent.

For two matrices A and B the min-plus product C is the matrix such that $C[i,j] = \min_k\{A[i,k] + B[k,j]\}$. Given an instance of min-plus product, nondeterministically guess C as well as a matrix K such that $K[i,j] = \mathrm{argmin}_k\{A[i,k] + B[k,j]\}$. We can easily check that $C[i,j] = A[i,K[i,j]] +$

$B[K[i,j],j]$ for all i and j, which proves that none of the entries in C are too large.

To verify none of the entries in C are too small, we construct a complete $n \times n \times n$ tripartite graph $G = (V_1, V_2, V_3, E)$ such that matrix $-A$ is the weight matrix for the edges between V_1 and V_2, $-B$ corresponds to the weights between V_2 and V_3, and C corresponds to the weights between V_1 and V_3. There are i, j, k such that $C[i,j] < A[i,k] + B[k,j]$ if and only if there is a negative weight triangle in this graph.

This reduction along with the co-nondeterministic algorithm for ZWT gives a nondeterministic algorithm for APSP with the claimed time complexity. □

Note that [33] in fact give a sizable list of problems that are equivalent to APSP under subcubic deterministic fine-grained reductions (including negative weight triangle, but not zero weight triangle). Our non-reducibility result therefore applies to all of these problems.

6. CHARACTERIZING THE QUANTIFIER STRUCTURE OF SETH-HARD GRAPH PROBLEMS

There are many problems within P that are known to be SETH-hard, but few of them are graph problems. And of the ones that are, they tend to have similar logical forms. For instance, k-Dominating Set [23] is definable by a $\forall^k\exists$ quantified formula; Graph Diameter-2 and Bipartite Graph Dominated Vertex [8] are definable by $\forall\forall\exists$ quantified formulas. Here we study the relations between SETH-hardness and the logical structures of model checking problems. The paper by Ryan Williams [28] explored the first-order graph properties on dense graphs, while in this paper, we look into sparse graphs whose input is a list of edges.

We define "graph property" quite broadly. The input to a graph property is a many-sorted universe that we view as sets of vertices, together with a number of unary relations (node colors), and binary relations, viewed as different categories or colors of edges. The binary relations can in general be directed. We specify the problem to be solved by a first order sentence. Let φ be a first order sentence in prenex normal form, which has k quantifiers.

$$\varphi = Q_1 x_1 \in X_1, Q_2 x_2 \in X_2, \ldots Q_k x_k \in X_k \psi$$

or shortened as

$$\varphi = Q_1 x_1 Q_2 x_2 \ldots Q_k x_k \psi$$

where φ is a quantifier-free formula whose atoms are unary or binary predicates on $x_1, \ldots, x_k$.

An instance of the model checking problem of φ gives k ($k \geq 3$) specifies sets $X_1, \ldots X_k$, where variable x_i is an element of set X_i, and unary or binary relations on these sets. (X_i needn't be disjoint, so allowing them to be viewed as distinct only increases the expressive power. We assume equality is one of the relations, so we can tell when $x_i = x_j$.) The sets $X_1, \ldots, X_k$ can be considered as the sets of nodes in a k-partite graph, and the values of a binary predicate can be considered as edges in the graph, i.e. for predicate P, $P(x_i, x_j) = true$ means there is an edge between nodes x_i and x_j. We refer to the k-partite graph with edges defined by predicate P as G_P, and the union of graphs defined on all predicates as G. The data structures used to code the relations are as follows: For each unary relation, an array of Booleans indexed by the vertices saying whether the relation holds, and for each binary predicate, the list representation of the corresponding directed graph. We want to see if φ is true for the input model.

Examples of this problem include k-Clique, which is defined by

$$\varphi = \exists x_1 \ldots \exists x_k \bigwedge_{i,j \in \{1,\ldots,k\}, i \neq j} E(x_i, x_j)$$

and k-Dominating Set, defined by

$$\varphi = \exists x_1 \ldots \exists x_k \forall x_{k+1} \left(E(x_1, x_{k+1}) \vee \cdots \vee E(x_k, x_{k+1})\right)$$

and Graph Radius-2, defined by

$$\varphi = \exists x_1 \forall x_2 \exists x_3 \left(E(x_1, x_3) \wedge E(x_3, x_2)\right)$$

We let $n = \max_i |X_i|$ be the maximum size of the node parts, and m be the number of edges in the union of the graphs. The size is $n + m$, but for convenience, we will assume $m > n$ and use m as the size.

The maximum deterministic complexity of a k-quantifier formula for $k \geq 2$ is $O(m^{k-1})$. For $k = 2$, this is just linear in the input size, so matching lower bounds follow. So the interesting case is $k \geq 3$. If SETH is true, some formulas require approximately this time. But if NSETH holds, all such formulas that are SETH hard are of the same logical form. This is made precise as follows:

Theorem 4. *Let $k \geq 3$. If* NSETH *is true, then there is a k-quantifier formula whose model checking problem is $O(m^{k-1})$* SETH*-hard, and all such formulas have the form $\forall^{k-1}\exists$ or $\exists^{k-1}\forall$.*

Theorem 4 comes directly from the following lemmas:

Lemma 12. *If* SETH *or* NSETH *is true, then there are $\forall^{k-1}\exists$ problems that are* SETH*-hard for time $O(m^{k-1})$.*

Thus by negating φ, the $\exists^{k-1}\forall$ problems are also hard under SETH.

On the other hand if a problem is of any form other than $\forall^{k-1}\exists$, we will show it has smaller nondeterministic complexity. Such a problem has either exactly one existential quantifier not in the innermost position, no existential quantifiers, or at least two existential quantifiers.

Lemma 13. *If φ has exactly one existential quantifier, but it is not on the innermost position, then it can be solved in $O(m^{k-2})$ nondeterminisitic time.*

Lemma 14. *If φ has more than one existential quantifiers, then it can be solved in time $O(m^{k-2})$ nondeterministically.*

These problems can be solved by guessing the existentially quantified variables, and exhaustive search on universally quantified variables. Because there are at most $k-2$ universial quantifiers, the algorithm runs in time $O(m^{k-2})$.

Lemma 15. *If all quantifiers are universal, then it can be solved in deterministic time $O(m^{k-1.5})$.*

Thus, only $\forall^{k-1}\exists$ formulas require $O(m^{k-1})$ nondeterministic time, and by looking at the complements, only $\exists^{k-1}\forall$ formulas require $O(m^{k-1})$ co-nondeterministic time. Thus, assuming NSETH, only these two types of first-order properties might be SETH-hard for the maximum difficulty of a k-quantifier formula.

Proofs of lemmas 12, 13 and 15 can be found in the full version of the paper.

7. CONSEQUENCES FOR VERIFICATION OF SOLUTIONS

Besides implying that some problems are not SETH-hard, NSETH also implies some new lower bounds on problems in P. Namely, if NSETH is true, then problems such as Fréchet distance, edit distance, and longest common substring also require quadratic co-nondeterministic time (i.e., to show that the optimal solution has cost that exceeds a given value). This immediately implies that, even given a solution, testing optimality requires quadratic time. We can formalize this as follows:

Theorem 5. *Let* $\mathrm{Opt}(x)$ *be the optimization problem, given* x*, find* $\max_{y,|y|=l(|x|)} F(x,y)$*, for some* F *that is computable in time* $T_F(n+l(n)) \geq n+l(n)$*. The verification problem* Ver *is: given* x *and* y*, is* y *an optimal solution for* Opt*, i.e., is there no* y' *with* $F(x,y') > F(x,y)$*. Assume that* Opt *is* SETH*-hard for some* $T(n)$ *which is greater than* $T_F^{1+\gamma}(n+l(n))$ *for some* $\gamma > 0$*. Then if* NSETH*,* Ver *cannot be solved in any time* T' *so that* $T_{\mathrm{Ver}}(n+l(n)) < T^{1-\epsilon}(n)$ *for any* $\epsilon > 0$*.*

Proof. Assume not, that Ver can be solved in some time T' with $T_{\mathrm{Ver}}(n+l(n)) < T^{1-\epsilon}(n)$. Then we can compute the function Opt in $\mathsf{NTIME}((T_{\mathrm{Ver}}(n+l(n)) + T_F(n+l(n))))$ as follows:

Non-deterministically guess an optimal solution y and run the algorithm for Ver on the pair (x,y). If it is optimal (i.e., in Ver), return $F(x,y)$. The total time is $l(n)$ to guess y, plus $T_F(n+l(n))$ to compute F, plus $T_{\mathrm{Ver}}(n+l(n))$.

From the assumption that Opt is SETH-hard for time $T(n)$, and since the time complexity of the above procedure is $O(T(n)^{1-\epsilon})$ for some $\epsilon > 0$, it follows that TAUT is in time $2^{n(1-\delta)}$ for some $\delta > 0$. This contradicts NSETH. □

So NSETH gives us a way to argue that not only finding but verifying optimal solutions is computationally intensive.

8. CONCLUSIONS AND OPEN PROBLEMS

A theme running through computational complexity is that looking at general relationships between models of computing and complexity classes can frequently shed light on the difficulty of specific problems. In this paper, we introduce this general technique to the study of fine-grained complexity by comparing nondeterministic complexities of problems. This raises the more general question of what other notions and models of complexity might be useful in distinguishing the fine-grained complexity of problems. For example, we show that neither 3-SUM or all-pairs shortest path can be SETH-hard if NSETH holds. This still leaves open the possibility that the two conjectures are equivalent to each other (if not to SETH). One might be able to prove such an equivalence, or give evidence against it by showing a different notion of complexity that distinguishes the two and is preserved by FGR.

9. ACKNOWLEDGMENTS

We would like to thank Amir Abboud, Karl Bringmann, Bart Jansen, Sebastian Krinninger, Virginia Vassilevska Williams, Ryan Williams and the anonymous reviewers for many helpful comments on an earlier draft.

This research is supported by NSF grant CCF-1213151 from the Division of Computing and Communication Foundations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

10. REFERENCES

[1] A. Abboud, A. Backurs, and V. V. Williams. Quadratic-time hardness of LCS and other sequence similarity measures. *CoRR*, abs/1501.07053, 2015.

[2] A. Abboud, V. Vassilevska Williams, and H. Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 41–50, New York, NY, USA, 2015. ACM.

[3] A. Abboud and V. V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014.

[4] V. V. W. Amir Abboud and J. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter. *arXiv preprint arXiv:1506.01799 (2015)*, pages 434–443, 2015.

[5] A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58, 2015.

[6] C. Beck and R. Impagliazzo. Strong ETH holds for regular resolution. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 487–494, 2013.

[7] R. Bellman. On a routing problem. Technical report, DTIC Document, 1956.

[8] M. Borassi, P. Crescenzi, and M. Habib. Into the square - on the complexity of quadratic-time solvable problems. *CoRR*, abs/1407.4972, 2014.

[9] K. Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. *CoRR*, abs/1404.1448, 2014.

[10] M. De Berg, M. M. de Groot, and M. H. Overmars. Perfect binary space partitions. *Computational Geometry*, 7(1):81–91, 1997.

[11] J. Edmonds. Maximum matching and a polyhedron with 0, l-vertices. *J. Res. Nat. Bur. Standards B*, 69(1965):125–130, 1965.

[12] L. R. Ford Jr. Network flow theory. Technical report, DTIC Document, 1956.

[13] A. Gajentaan and M. H. Overmars. On a class of o (n 2) problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

[14] J. E. Hopcroft and R. M. Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[15] R. Impagliazzo and R. Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001.

[16] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512 – 530, 2001.

[17] H. Jahanjou, E. Miles, and E. Viola. Local reductions. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:99, 2013.

[18] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 217–226. SIAM, 2014.

[19] M. Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14(3):205–220, 1967.

[20] S. Micali and V. V. Vazirani. An o (v| v| c| e|) algoithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 17–27. IEEE, 1980.

[21] D. Moeller, R. Paturi, and S. Schneider. Subquadratic algorithms for succinct stable matching. 2015.

[22] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations research*, 41(2):338–350, 1993.

[23] M. Patrascu and R. Williams. On the possibility of faster SAT algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075, 2010.

[24] P. Pudlak and R. Impagliazzo. A lower bound for dll algorithms for k-sat. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2000, San Francisco, CA, USA, January 9-11, 2000*, pages 128–136, 2000.

[25] V. Vassilevska and R. Williams. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 455–464. ACM, 2009.

[26] R. Williams. A new algorithm for optimal constraint satisfaction and its implications. *Electronic Colloquium on Computational Complexity (ECCC)*, (032), 2004.

[27] R. Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

[28] R. Williams. Faster decision of first-order graph properties. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 80:1–80:6, 2014.

[29] R. Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014.

[30] R. Williams. Personal communication. 2015.

[31] V. V. Williams. Stoc tutorial: Hardness and equivalences in p. `http://theory.stanford.edu/~virgi/stoctutorial.html`.

[32] V. V. Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *LIPIcs-Leibniz International Proceedings in Informatics.*, volume 43, 2015.

[33] V. V. Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 645–654. IEEE, 2010.