On polynomial lower bounds based on complexity of SAT problems

May 7, 2016

Abstract

A large amount of recent work in complexity theory seeks to prove tight lower bounds on problems within P under the Strong Exponential Time Hypothesis (SETH) and similar conjectures. Lower bounds for many problems can be proved via reduction from CNF formulas under SETH. More recently, work by Abboud, Hansen, Williams and Williams [3] proves bounds under reductions from more complex representations of SAT; and Abboud, Vassilevska and Yu [5] prove bounds conditional on at least one out of three separate conjectures.

We explore three such reductions to Edit Distance [7] and Longest Common Subsequence (LCS) [1]. Currently, no strongly subquadratic time algorithms exist to compute these problems. If any of the assumptions made in these papers are true, the current near-quadratic time bounds can be shown to be tight. We also explore a proof of conditional lower bounds on various types of graph problems by Abboud et al. and compare the efficacy of these different methods for proving conditional lower bounds within P.

1 Introduction

Edit-Distance, the number of operations needed to convert one string to another, is an incredibly important computation problem, whose common dynamic programming algorithm runs in time $O(n^2)$. This runtime is prohibitive in of the practical applications of Edit Distance, many of which lie in computational biology or genomics with input sizes on the order of billions. While faster approximation algorithms have been developed, there has been no success thus far in proving upper bounds below $O(n^2/\log n)$ for this problem. Current bounds on Longest Common Subsequence (LCS) are identical to those for Edit-Distance [3].

String manipulation is not the only area in which lower bounds are highly sought after. Many questions in graph theory have no known algorithms that run in time less than $O(n^c)$ for some constant c. These include Δ-Matching-Triangles (best running-time n^3), amortized updates and queries for Max-Flow and other reachability problems (n^1), and a slew of other graph problems with running times of n^2 or n^3. Proving lower bounds on these problems is beyond the scope of current complexity theory.

Due to the difficulty that has presented itself in proving unconditional polynomial lower bounds for problems within P (or NP, for that matter), many researchers have turned instead to proving lower bounds conditional on certain conjectures. Chief among these is the Strong Exponential Time Hypothesis, or SETH, introduced by Impagliazzo and Paturi [10]. It states that solving k-SAT on CNF formulas will take almost $O(2^n)$ time. In a significant recent paper, Backurs and Indyk show that Edit-Distance [7] and LCS [1] do not have strongly subquadratic time algorithms if SETH is true.

Lower bounds for many problems beyond merely those stated above have been proved under SETH. However, it is a very strong assumption that many researchers believe to be false. In this survey, we also discuss two further extensions to this current line of research. First, a reduction from the satisfiability of Branching Programs presented by Abboud et al. that proves lower bounds for Edit-Distance and LCS under weaker assumptions than SETH [3]. Second, results from Abboud et al. that prove lower bounds on several graph problems contingent on one of three conjectures being true: SETH, the 3-SUM conjecture, and the APSP conjecture [5]. The 3-SUM and APSP conjectures posit lower bounds on the 3-SUM problem of n^2 and on the all pairs shortest-path problem of n^3, respectively.

We introduce all these conjectures in detail in Section 2. In Section 3, we formally define the problems to which we show conditional lower bounds. Most of the main theorems and conjectures will be presented in these two sections. We give proofs to most of these in Section 4, where we actually present the reductions.

2 Conjectures

2.1 The Exponential Time Hypothesis

At a glance, the Exponential Time Hypothesis (ETH) is the hypothesis that solving 3-SAT takes worst case exponential time. Notice that this is a stronger claim than P != NP, because there exist functions between polynomial and exponential (e.g. $2^{\sqrt{n}}$). The Strong Exponential Time Hypothesis (SETH) claims that solving k-SAT takes almost $O(2^n)$ time, i.e. you cannot do much better than checking all 2^n assignments of variables.

We will now rigorize these conjectures [9]. For each integer $k \geq 2$, define a real number s_k to be the infimum of the real numbers δ for which there exists an algorithm solving k-SAT in time $O(2^{\delta n})$, where n is the number of variables in the given k-SAT instance.

Note that $s_2 = 0$, because 2-SAT can be solved in polynomial time.

Conjecture 1. *(ETH) For every $k > 2$, $s_k > 0$.*

Clearly, $s_3 \leq s_4 \leq \ldots$, so it is equivalent to assume that $s_3 > 0$ the positivity of the remaining numbers s_k follows automatically from this assumption.

Conjecture 2. *(SETH) The limiting value s_∞ of the sequence of numbers s_k equals one.*

2.2 The SAT hierarchy as an Extension to SETH

While all representations of SAT are equivalent in terms of whether or not they can be solved with a polynomial-time algorithm, the finer-grained complexity of SAT problems differs based on the complexity and obscurity of the representation. A major weakness for SETH is that it relies on the difficulty of satisfying CNF formulas, which lie low on the hierarchy of complexity of representation.

Abboud et al. [3] present an extension to SETH, $\mathcal{C}$-SETH, that makes similar claims, but for an arbitrary class $\mathcal{C}$ of SAT representations.

Conjecture 3. *($\mathcal{C}$-SETH) Given a representation class $\mathcal{C}$, satisfiability of an input in $\mathcal{C}$ cannot be computed in time $(2-\epsilon)^n$ for any $\epsilon > 0$.*

As the complexity of the class $\mathcal{C}$ increases, the assumption on the bounds of $\mathcal{C}$-SETH becomes weaker, and Conjecture 3 becomes more credible. Proving lower bounds conditional on $\mathcal{C}$-SETH is therefore appealing for richer representations such as BP-SAT, which we will present below, as the consequences of a strongly sub-quadratic time algorithm for Edit-Distance would be farther reaching.

2.3 The 3-SUM and APSP Conjectures

The 3-SUM and APSP conjectures posit that the current upper bounds on algorithms for the 3-SUM and all pairs shortest path problems are tight.

Conjecture 4. *(The 3-SUM Conjecture) There is no algorithm that can check whether a list of n numbers contains three that sum to zero (the 3-SUM problem) in $O(n^{2-\epsilon})$ time for any $\epsilon > 0$.*

Conjecture 5. *(The APSP Conjecture) There is no algorithm that can compute all pairs shortest paths (APSP) on n node weighted graphs in $O(n^{3-\epsilon})$ time for any $\epsilon > 0$.*

As with SETH and $\mathcal{C}$-SETH, these conjectures are largely based on empirical arguments. Like Edit-Distance and LCS, 3-SUM and APSP are well-studied in their respective fields. Despite the effort that has been made to find faster algorithms for either problem, no strongly subquadratic or subcubic formulas have been found for 3-SUM or APSP, respectively.

Conjecture 6. *At least one of the 3-SUM conjecture, the APSP conjecture, or SETH, is true.*

As an amalgam of three seemingly unrelated conjectures, the above conjecture may seem odd; however, this is actually what makes it so popular in the field of complexity theory. Any one conjecture on its own may seem refutable, but that researchers in the fields of computational geometry, graphs algorithms and exact algorithms all have separately failed to find faster algorithms for these important problems is far less likely. Thus proving lower bounds conditional on the Conjecture 6 above becomes very attractive [5].

3 Introduction to Relevant Problems

3.1 Definitions of Problems

Before presenting our reductions under the above conjectures, we define the main problems for which we will prove conditional lower bounds. These problems fall into two categories: distance measures and various graph problems.

3.1.1 Distance Measures

We will begin by providing precise definitions of various distance measures.

Definition 1. (Edit Distance) For any two sequences x and y over an alphabet Σ, the edit distance $EDIT(x, y)$ is equal to the minimum number of symbol insertions, symbol deletions or symbol substitutions needed to transform x into y.

Remark 1. Some definitions of edit distance are more general, and assign different weights to substitutions, insertions, and deletions. In the rest of this paper, we will not distinguish between different types of edits.

Definition 2. (Longest Common Subsequence) For any two sequences x, y over an alphabet Σ, the longest common subsequence $LCS(x, y)$ is equal to the length of the longest sequence that appears as a subsequence in both input strings.

3.1.2 Orthogonal Vectors

The orthogonal vectors problem will be our starting point for proving conditional lower bounds on Edit-Distance and LCS. We will prove that there is no strongly subquadratic time algorithm to find orthogonal vectors unless SETH is false.

Definition 3. (Orthogonal Vectors) Given two lists $\{\alpha_i\}_{i\in[n]}$ and $\{\beta_i\}_{i\in[n]}$ of vectors $\alpha_i, \beta_i \in \{0,1\}^d$ and an integer $r \in \{0, \dots, d\}$, is there a pair α_i, β_j that are orthogonal?

Definition 4. (Most Orthogonal Vectors) Given two lists $\{\alpha_i\}_{i\in[n]}$ and $\{\beta_i\}_{i\in[n]}$ of vectors $\alpha_i, \beta_i \in \{0,1\}^d$ and an integer $r \in \{0, \dots, d\}$, is there a pair α_i, β_j that has inner product at most r? We call any two vectors that satisfy this condition (r-)far, and (r-)close vectors otherwise.

Note that this most orthogonal vectors is harder than the regular Orthogonal Vectors problem.

The reductions to Edit Distance and LCS will be reductions from the Most-Orthogonal vectors problem. The dependency of SETH on the quadratic lower bounds on EDIT and LCS is stated in the following three theorems:

Theorem 1. *[7] If there exists an algorithm for Most Orthogonal Vectors that runs in $O(2^{n-\epsilon})$ time, then there is an algorithm for k-SAT that runs in $O(2^{n-2\epsilon} poly(n))$ time (which contradicts SETH).*

Theorem 2. *If EDIT can be computed in time $O(n^{2-\delta})$ for some $\delta > 0$ on two sequences of length n over an alphabet of size 4, then Orthogonal Vectors Problem with $|A| = |B| = N$ and $A, B \subset \{0,1\}^d$ can be solved in time $d^{O(1)} \cdot N^{2-\delta}$.*

Theorem 3. *If LCS can be computed in time $O(n^{2-\delta})$ for some $\delta > 0$ on two sequences of length n over an alphabet of size 4, then Orthogonal Vectors Problem with $|A| = |B| = N$ and $A, B \subset \{0,1\}^d$ can be solved in time $d^{O(1)} \cdot N^{2-\delta}$.*

3.1.3 BP-SAT

In addition to reductions under SETH, we will prove lower bounds on LCS under BP-SETH [3]. We first define the satisfiability of Branching Programs, or BP-SAT. A nondeterministic Branching Program of length T and width W is defined as a graph with T layers of W nodes each, with a single start node u_{start} at the first layer, and the single accept node u_{accept} at the last layer. All the nodes in a given layer except the last one are marked with a variable x_i, and have an arbitrary number of outgoing edges to the next layer, each marked 0 or 1. The BP is evaluated on an input $x_1, x_2, ..., x_n \in \{0, 1\}$ as follows: starting at u_{start}, at each node $u_{i,j}$ marked with variable x_k, we can traverse any edge e marked with value $c \in \{0, 1\}$ such that $c = x_k$. The BP accepts iff it terminates at node u_{acc}.

Definition 5. (BP-SAT) Given a Branching Program P on n boolean inputs, decide if there is an assignment to the variables that makes P accept.

We will present a reduction from BP-SAT in the same vein as Backurs' reduction from CNF-SAT [7] which shows the following to be true.

Theorem 4. *There is a reduction from SAT on nondeterministic branching programs on n variables, length T, and width W, to an instance of Edit-Distance or LCS on two binary sequences of length $N = 2^{n/2} \cdot T^{O(\log W)}$, and the reduction runs in $O(N)$ time.*

Equivalently, the theorem above implies that if there is a strongly sub-quadratic algorithm for Edit-Distance or LCS, BP-SETH is false. This has much stronger consequences than if SETH were to be refuted. Theorem 4 has the following corollary.

Corollary 5. *[3] If Edit Distance and LCS on two binary sequences of length N is in $O(N^{2-\epsilon})$ time for some $\epsilon > 0$, then the complexity class E^{NP} does not have:*

1. *non-uniform $2^{o(n)}$-size boolean circuits*
2. *non-uniform $o(n)$-depth circuits of bounded fan-in*
3. *non-uniform $2^{o(n^{1/2})}$-size nondeterministic branching programs*

3.1.4 Matching Triangles

The Δ-Matching-Triangles problem and the Triangle-Collection problem address the existence of tri-colored triangles within graphs. Both problems can serve as the starting point for reductions that extend to answer many core questions in graph theory.

Definition 6. (Triangle-Collection) Given a graph G with colored nodes, is it true that for all triples of distinct colors (a, b, c) there is a triangle (x, y, z) in G in which x has color a, y has color b, and z has color c?

Definition 7. (Δ-Matching-Triangles) Given a graph G with colored nodes, is there a triple of distinct colors (a, b, c) such that there are at least Δ triangles (x, y, z) in G in which x has color a, y has color b, and z has color c? (Are there Δ triangles with "matching" colors?)

The $O(n^3)$ algorithm for these problems is trivial. However, no strongly subcubic time algorithm for either has been found. With these definitions we have the following theorem.

Theorem 6. *Conjecture 6 implies that Triangle-Collection and Δ-Matching-Triangles, with $\omega(1) < \Delta < n^{o(1)}$, on graphs with n nodes cannot be solved in $O(n^{3-\epsilon})$ time, for any $\epsilon > 0$ [5].*

Not only does the above theorem give us a lower bound on important problems in graph theory, but the tightness of the lower bound allows for efficient reductions to other, perhaps better-known problems. This theorem gives potentially tight lower bounds for many problems within graph theory conditional on a relatively weak conjecture. We give a proof of the reduction to Δ-Matching-Triangles in 4.5.1.

3.1.5 Dynamic Graph Problems

We will explore further reductions from Triangle-Collection conditional on these lower bounds to prove lower bounds for a large number of dynamic problems, including Single-Source-Reachability (#SSR), Strongly Connected Components (#SCC), Subgraph Connectivity (#SS-Sub-Conn), and Max-Flow.

Definition 8. (#SSR) Maintain a directed graph under insertions and deletions of edges, and queries on whether there is a path from a given node s to some node t

Definition 9. (#SCC) Maintain a directed graph under insertions and deletions of edges, and queries on the number of strongly connected components.

Definition 10. (#SS-Sub-Conn) Maintain a directed graph and support turning nodes of and on, and querying whether there is a path from s to t containing only "on" nodes.

Definition 11. (Max-Flow) Maintain a directed graph under reweighting of edges and querying of the maximum possible flow.

Several previous attempts have been made to prove conditional lower bounds for these problems. A $O(n^{\frac{2}{3}-o(1)})$ lower bound has been proved under 3-SUM [11]. However, it is not known if this bound is tight, as it does not match current upper bounds. Indeed, lower bounds of $n^{1-o(1)}$ have been proven under SETH [6]. Higher lower bounds under 3-SUM using the aforementioned approach seem unlikely, and no bounds have been shown previously under APSP. Abboud et al. present new reductions from the Triangle-Conjecture under Conjecture 6 for these problems [5].

Theorem 7. *Conjecture 6 implies that any dynamic algorithm of #SSR, #SSC, #SS-Sub-Conn, and Max-Flow requires either amortized $n^{1-o(1)}$ update or query times, or $n^{3-o(1)}$ processing time.*

Lower bounds for static variants on Max-Flow can also be proved using this approach, although for the sake of space we do not present these here [5].

3.2 Existing Algorithms

3.2.1 An algorithm for Edit Distance

Edit Distance between two strings of length n can be computed in $O(n^2)$ time using elementary algorithms. For example, here is an algorithm that reduces the problem to the shortest path problem:

- On input x, y, define a subproblem $X(i, j)$ to be the answer edit distance between $x[i : n]$ and $y[j : n]$
- Construct a graph on the lattice $[0...n] \times [0...n]$, such that an edge from point a to point b corresponds to the assertion that $X(a) \leq X(b) + 1$.
- Add all edges $(i, j) \rightarrow (i + 1, j)$, $(i, j) \rightarrow (i, j + 1)$.
- For all i, j such that $x[i] = y[j]$, add an edge $(i, j) \rightarrow (i + 1, j + 1)$.
- Compute the shortest path from $(0, 0)$ to (n, n). This is the answer to $X(0, 0)$.

The proof of correctness of this algorithm is omitted, but here is a brief sketch: Consider a sequence of edits to x that ends at y. Let it be sorted. Then the lengths of the unmatched suffixes of x, y will correspond to subproblems as defined in the algorithm. Therefore any sequence of edits corresponds to a path through the above graph.

3.2.2 An algorithm for LCS

Longest Common Subsequence can be computed in $O(n^2)$ time using a similar algorithm, where edges instead correspond to either assertions that $X(i, j) \geq X(i, j+1)$, $X(i, j) \geq X(i+1, j)$, or $X(i, j) \geq 1+X(i+1, j+1)$. The problem can then be solved by evaluating X on all (i, j), in topological order (e.g. decreasing i, j), starting with $X(n, n) = 0$.

3.2.3 An algorithm for Most Orthogonal Vectors

Most orthogonal vectors can be solved in $O(n^2)$ time using the trivial algorithm, i.e. compute the inner product of every pair $(\alpha_i, \beta_j) \in \alpha \times \beta$, and check if it is less than r.

3.2.4 An algorithm for Matching Triangles

Both the Triangle-Collection and Δ-Matching-Triangles problems can be solved in $O(n^3)$ time on input graph $G = (V, E)$ with coloring χ by checking each triple (x, y, z) of nodes in V.

4 Reductions

4.1 k-SAT to Most Orthogonal Vectors

We will prove Theorem 1.

Theorem 8. *If Most-Orthogonal Vectors on n vectors in $\{0,1\}^d$ can be solved in $T(n,d)$ time, then given a CNF formula on n variables and M clauses, we can compute the maximum number of satisfiable clauses (MAX-CNF-SAT), in $O(T(2^{n/2}, M) \cdot \log M)$ time.*

Proof. [7] Given a CNF formula on n variables and M clauses, split the variables into two sets of size $n/2$ and list all $2^{n/2}$ partial assignments to each set. Define a vector $v(\alpha)$ for each partial assignment α which contains a 0 at coordinate $j \in [M]$ if α sets any of the literals of the j-th clause of the formula to true, and 1 otherwise. In other words, it contains a 0 if the partial assignment satisfies the clause and 1 otherwise.

Now, observe that if α, β are a pair of partial assignments for the first and second set of variables, then the inner product of $v(\alpha)$ and $v(\beta)$ is equal to the number of clauses that the combined assignment (α, β) does not satisfy. Therefore, to find the assignment that maximizes the number of satisfied clauses, it is enough to find a pair of partial assignments α, β such that the inner product of $v(\alpha), v(\beta)$ is minimized. The latter can be easily reduced to $O(\log M)$ calls to an oracle for Most-Orthogonal Vectors on $N = 2^{n/2}$ vectors in $0, 1^M$ with a standard binary search. $\square$

Theorem 1 follows immediately.

4.2 Most Orthogonal Vectors to Edit Distance

The reduction was provided in *Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false)* [7]. We will outline the reduction here.

4.2.1 Coordinate gadgets

We will first define two coordinate gadgets, CG_1, CG_2 as follows:

$$CG1(x) := \begin{cases} (0^{l_1} 0^{l_0} 1^{l_0} 1^{l_0} 1^{l_0} 0^{l_1} & \text{if } x = 0 \\ (0^{l_1} 0^{l_0} 0^{l_0} 0^{l_0} 1^{l_0} 0^{l_1} & \text{if } x = 1 \end{cases}$$

$$CG_2(x) := \begin{cases} (0^{l_1} 0^{l_0} 0^{l_0} 1^{l_0} 1^{l_0} 0^{l_1} & \text{if } x = 0 \\ (0^{l_1} 1^{l_0} 1^{l_0} 1^{l_0} 1^{l_0} 0^{l_1} & \text{if } x = 1 \end{cases}$$

These coordinate gadgets have the property that

$$EDIT(CG_1(x_1), CG_2(x_2)) = \begin{cases} l_0 & \text{if } x_1 \cdot x_2 = 0 \\ 3l_0 & \text{otherwise} \end{cases}.$$

The coordinate gadgets will be used to construct vector gadgets.

4.2.2 Vector gadgets

We will define two vector gadgets, VG_1 and VG_2, as follows:

$$VG_1(a) = Z_1 \circ L \circ V_0 \circ R \circ Z_2, \quad VG_2(b) = V_1 \circ D \circ V_2,$$

where

$$Z_1 = Z_2 = 0^{l_2}, \quad V_1 = V_2 = V_0 = 1^{l_2}.$$

These vector gadgets have the property that

$$EDIT(VG_1(x_1), VG_2(x_2)) : \begin{cases} \leq E_s & \text{if } x_1 \cdot x_2 = 0 \\ = E_u & \text{otherwise} \end{cases}.$$

4.2.3 The reduction: Sequences of gadgets

We will now, given sets $\{\alpha\}_i$ and $\{\beta\}_i$, construct a sequence of gadgets as follows: Let $t = \max(|VG_1|, |VG_2|)$ and $T = 1000d \cdot t$ Define $VG'_k(x) = 2^T VG_k(x) 2^T$ for $k \in \{1, 2\}$, and let $f \in \{0,1\}^d$ be a vector with all 1s.

Finally, let

$$\begin{aligned} P_1 &= VG'_1(\alpha_1) \circ \ldots \circ VG'_1(\alpha_n), \\ P_2 &= VG'_2(f)^{|A|-1} \circ VG'_2(\beta_1) \circ \ldots \circ VG'_2(\beta_n) \circ VG'_2(f)^{|A|-1}. \end{aligned}$$

The construction ensures that, if there exists i, j so that $\alpha_i \cdot \beta_j = 0$, then $EDIT(P_1, P_2) \leq X - (E_u - E_s)$, and otherwise, $EDIT(P_1, P_2) = X$, for some constant X.

Here is a sketch of the proof:

Proof. First, we claim that in the optimal alignment (i.e. isolate the edits that are substitutions, and look at which characters of P_1 become characters in P_2), each VG'_1 gadget will be aligned (i.e., matched) with some VG'_2 gadget (due to the 2^T padding of VG'_k). The $|B| + |A| - 2$ unmatched VG'_2 gadgets will be deleted.

Each match α_i, β_j contributes $EDIT(VG'_1(\alpha_i), VG'_2(\beta_j))$ to the total cost. However, $EDIT(VG'_1(\alpha_i), VG'_2(\beta_j) = EDIT(VG_1(\alpha_i), VG_2(\beta_j))$. Recalling the property of vector gadgets stated in 4.2.2, it is now clear that the matching has total cost at most $X - (E_u - E_s)$ if and only if it matches some α_i, β_j with $\alpha_i \cdot \beta_j = 0$.

Finally, given that some orthogonal α_i, β_j, it is possible to construct a matching in which they are matched together:

- Match $\alpha_1 \ldots \alpha_{i-1}$ with $i - 1$ of the head sequence of $VG'_2(f)$
- Match α_i with β_j
- Match $\alpha_{i+1} \ldots \alpha_n$ with $n - i$ of the tail sequence of $VG'_2(f)$.

This is always possible, as the head and tail sequences of $VG'_2(f)$ have length $|A| - 1$ each. □

4.3 Most Orthogonal Vectors to LCS

The following reduction is from *Quadratic-Time Hardness of LCS and other Sequence Similarity Measures*[1] We reduce to weighted LCS (WLCS), where each symbol has a weight, and the goal is to pick a common subsequence with maximum weight. It is easy to reduce WLCS to LCS.

4.3.1 Coordinate gadgets

Let the alphabet be $\Sigma = \{0, 1, 2, 3, 4, 5, 6\}$. Give weights $w(4) = w(6) = 1$, $w(5) = X = 100d$.

Define

$$CG_1(\alpha, i) = \begin{cases} 5465 & \text{if } \alpha[i] = 0 \\ 545 & \text{otherwise} \end{cases}$$
$$CG_2(\beta, j) = \begin{cases} 5645 & \text{if } \beta[j] = 0 \\ 565 & \text{otherwise} \end{cases}$$

These coordinate gadgets have the property that

$$LCS(CG_1(\alpha, i), CG_2(\beta, i)) = \begin{cases} 2X + 1 & \text{if } \alpha[i] \cdot \beta[j] = 0 \\ 2X & \text{otherwise} \end{cases}.$$

4.3.2 Vector gadgets

First, define R_1, R_2 to be the concatenation of coordinate gadgets.

$$R_1(\alpha) = CG_1(\alpha, 1) \circ \ldots \circ CG_1(\alpha, n)$$
$$R_2(\beta) = CG_G(\beta, 1) \circ \ldots \circ CG_1(\beta, n)$$

Note that $WLCS(R_1(\alpha), R_2(\beta)) = d \cdot 2X + (d - \alpha \cdot \beta)$. Then, let

$$VG_1 = 1 \circ R_1(\alpha)$$
$$VG_2 = R_2(\beta) \circ 1,$$

and give the weight $w(1) = A = d \cdot 2x + (d - (r - 1))$. Note that 1 is always a common subsequence, so $WLCS(VG_1, VG_2) \geq A$.

These vector gadgets have the property that

$$WLCS(VG_1(\alpha), VG_2(\beta)) : \begin{cases} \geq A + 1 & \text{if } \alpha \cdot \beta \leq r \\ = A & \text{otherwise} \end{cases}.$$

4.3.3 The reduction: Sequences of gadgets

Let $VG_1'(\alpha) = 0 \circ VG_1(\alpha) \circ 2$ and $VG_2'(\beta) = 0 \circ VG_2(\beta) \circ 2 \circ 3$.

Let $\alpha_1 \ldots \alpha_n$ and $\beta_1 \ldots \beta_n$ be the given sets of vectors. Let f be a vector of all ones. Then, let

$$P_1 = 3^{|P_2|} \circ VG_1'(\alpha_1) \circ \ldots \circ VG_1'(\alpha_n) \circ 3^{|P_2|}$$
$$P_2 = 3VG_2'(f)^{n-1} \circ VG_2'(\beta) \circ \ldots \circ VG_2'(\beta) \circ VG_2'(f)^{n-1}.$$

Set the weights $w(3) = B = A^2, w(0) = w(2) = C = B^2$. Then, these sequences have the property that

$$WLCS(P_1, P_2) : \begin{cases} \geq n(2C + A) + 2nB + 1 & \text{if } \alpha_i \cdot \beta_j \leq r \\ \leq n(2C + A) + 2nB & \text{otherwise} \end{cases}.$$

Thus, it suffices to solve $WLCS(P_1, P_2)$ to determine if there exist α_i, β_j that are r-close.

Because $|P_1|, |P_2|$ are linear in n, we conclude that if a strongly subquadratic algorithm exists for LCS, then a strongly subquadratic algorithm exists for Most Orthogonal Vectors.

4.4 Branching Programs to Edit-Distance and LCS

In *Simulating Branching Programs with Edit Distance and Friends* [3], Abboud et al. introduce a new reduction to Edit Distance and LCS that shows there is no strongly subquadratic algorithm under much weaker assumptions than SETH. Specifically, they define a reduction from the satisfiability of nondeterministic branching programs to instances of either Edit Distance or LCS. They thus prove that a strongly subquadratic algorithm for LCS would imply that BP-SAT can be computed in time $(2-\delta)^n$ time. This is an important departure from previous work in this vein ([1], [7]), which gives reductions from k-SAT under the original SETH. Unlike CNF formulas, which are fairly low in the hierarchy of richness of representation, BPs are able to represent complex concepts such as linear-algebraic operations and cryptographic primitives. The reductions presented here are much more consequential. Abboud et al. [3] show that even mildly subquadratic algorithms for LCS would have a large impact on lower bounds for SAT.

4.4.1 BP-SAT to LCS

The full reduction is quite long, but for the sake of this paper we present a shorter version. Given a branching program of length T and width W, a reduction to LCS with an alphabet of size $|\Sigma| = O(W \log T)$ simplifies the presentation while maintaining the spirit of the proof. However, a more robust reduction with $|\Sigma| = 2$ can be found in the original paper [3].

Similar to the reduction from k-SAT to LCS, we define an intermediary problem. On an input to BP-SAT on n variables, split the variables into two sets $X_1 = \{x_1, ..., x_{n/2}\}$ and $X_2 = \{x_{n/2+1}, ..., x_n\}$, and consider whether there exist partial assignments $a \in \{0,1\}^{n/2}$ and $b \in \{0,1\}^{n/2}$ to these variables, respectively, such that the original BP is satisfied. Solving BP-SAT is equivalent to determining whether such a pair a, b exists.

Theorem 9. *There is a constant c such that if LCS can be solved in time $S(N)$, then BP-SAT on n variables and programs of length T and width W can be solved in $S(2^{n/2} \cdot T^{c \log W})$ time.*

Given a branching program P, and the corresponding function F, split the input variables into X_1 and X_2. Deciding BP-SAT is equivalent to determining if there exist some $a, b \in \{0,1\}^{n/2}$ such that $F(a \circ b) = 1$. As with the previous reductions from orthogonal vectors to LCS, we construct gadget sequences $G(a)$ and $\overline{G}(b)$ for every a, b such that for some integer Y, we have the following.

$$LCS(G(a), \overline{G}(b)) \begin{cases} = Y & \text{if } F(a \circ b) = 1 \\ \leq Y - 1 & \text{otherwise} \end{cases}$$

We then combine $G(a)$ for all $a \in \{0,1\}^{n/2}$ into some A, and $\overline{G}(b)$ for all $b \in \{0,1\}^{n/2}$ into some B, such that for some integer E we have

$$LCS(A,B) \begin{cases} = E & \text{if } F(a \circ b) = 1 \text{ for some } a, b \\ \leq E - 1 & \text{otherwise} \end{cases}$$

We call on a lemma from Abboud et. al [2] to prove the existence of A and B.

Lemma 10. *Take function $F\{0,1\} \to \{0,1\}$. If, given any $a, b \in \{0,1\}^{n/2}$, one can construct gadget sequences $G(a)$, $\overline{G}(b)$ of length L, L' respectively such that for some integer Y,*

$$LCS(G(a), \overline{G}(b)) \begin{cases} = Y & \textit{if } F(a \circ b) = 1 \\ \leq Y - 1 & \textit{otherwise} \end{cases}$$

Then, one can construct sequences A, B such of length $2^{n/2} poly(L, L')$ such that for some integer E,

$$LCS(A,B) \begin{cases} = E & \textit{if } F(a \circ b) = 1 \textit{ for some } a, b \\ \leq E - 1 & \textit{otherwise} \end{cases}$$

Thus to prove Theorem 8, given branching program P of length T and width W, we need only construct gadget sequences G, $\overline{G}$ of length $T^{O(\log W)}$ that satisfy the above property. Let $T = 2^t + 1$ for simplicity for some $t \geq 0$. Let $G(a)$ and $\overline{G}(b)$ represent the subsets of edges in P; our goal is to decide if there is some path from u_{start} to u_{acc}. We can do so by guessing, recursively, which node this path passes through in level $2^{t-k} + 1$ for integer k, $k \geq 1$, and splitting the BP into two BPs of half the length.

At each recursion level we have two nodes $u \in L_i$ and $v \in L_j$, $j - i = 2^k$, and we want to decide if there is some path from u to v. Denote the sequences for a and b by $RG_k^{u,v}(a)$ and $\overline{RG}_k^{u,v}(b)$ respectively. We define these sequences such that for some Y_k,

$$LCS(RG_k^{u,v}(a), \overline{RG}_k^{u,v}(b)) \begin{cases} = Y_k & \text{if on input } a \circ b, v \text{ is reachable from} \\ & u \text{ in } 2^k \text{ steps} \\ \leq Y - 1 & \text{otherwise} \end{cases}$$

We have a base case $k = 0$, u and v are connected iff there is an edge from u to v. The variable $x(i)$ that corresponds to the existence of this edge must be contained in either a and b; whichever sequence contains $x(i)$ is assigned e if an edge is present, and the other is assigned e regardless. Thus $Y_0 = 1$. For $k > 0$, construct $RG_k^{u,v}(a)$ and $\overline{RG}_k^{u,v}(b)$ from each of the W choices of $RG_{k-1}^{u,y}(a)$ and $\overline{RG}_{k-1}^{y,v}(b)$, where y is some node on the layer in the middle of u and v. We have $G(a) = RG_t^{u_{start}, u_{acc}}(a)$ and $G(b) = \overline{RG}_t^{u_{start}, u_{acc}}(b)$, and have $F(a \circ b) = 1$ iff $LCS(G(a), G(b)) = T$.

Since each gadget will have length at most $W^{O(t)}$, we have sequences of at most $2^{n/2} \cdot T^{O(\log W)}$ by Lemma 10, completing the proof of Theorem 9.

4.5 Matching Triangles Problem

In *Matching triangles and basing hardness on an extremely popular conjecture*, Abboud et al. [5] attempt a different approach to conditional lower bounds in showing that the currently conjectured bounds hold for several graph problems if any one of three conjectures hold. Namely, the 3-SUM conjecture, the APSP conjecture, and SETH (see Conjecture 6).

4.5.1 3-SUM and APSP to Matching Triangles

We first show a reduction from 3-SUM and APSP via an intermediate problem, EW-Triangle.

Definition 12. (EW-Triangle) Given a graph $G = (V, E)$ with integer edge weights $w : E \to [-n^c, n^c]$, determine if there is a triangle (a, b, c) of total weight $w(x, y) + w(x, z) + w(y, z) = 0$.

EW-Triangle cannot be solved in time less than $n^{3-o(1)}$ unless both 3-SUM [12] and APSP [13] are false. The basic idea in the reduction from EW-Triangles is to produce a set of $n^{o(1)}$ mappings from integers $[-n^c, n^c]$ to vectors in $[-p, p]^d$, where $(p/3)^d > n^c$ so that three numbers sum to zero if and only if the three corresponding vectors sum to some target vector t.

Lemma 11. *For any integers $n, c, d, p \geq 1$ such that $p \geq 3n^{\lceil c/d \rceil}$, there is a set of $s = 2^{O(d)}$ mappings $f_1, ..., f_s : [-n^c, n^c] \to [-p/3, p/3]^d$ and s target vectors $t_1, ..., t_s \in [-p, p]^d$ such that for any three numbers $x, y, z \in [-n^c, n^c] : x+y+z = 0$ if and only if, for some $i \in [s]$, $f_i(x) + f_i(y) + f_i(z) = t_i$.*

Full proof of Lemma 11 can be found in a separate paper by Abboud, Lewi and Williams [4]. From this lemma we are able to define the following reduction from EW-Triangle to Δ-Matching-Triangles.

Lemma 12. *An instance of EW-Triangle on n nodes, m edges, and edge weights in $[-n^c, n^c]$ can be reduced to $s = 2^{O(\Delta)}$ instances of Matching-Triangles on $O(n \cdot n^{c/\Delta} \cdot \Delta)$ nodes and $O(mn^{c/\Delta}\Delta)$ edges in linear time.*

Proof. On input $G = (V, E)$, $V = A \cup B \cup C$, with edge weights in $[-n^c, n^c]$ to EW-Triangle, construct unweighted graph $G' = (V', E')$ with $O(n \cdot n^{c/\Delta} \cdot \Delta)$ nodes and coloring $\chi : V_i' \to [n]$ as follows. □

From Lemma 11, we take $d = \Delta$, $p = O(n^{\lceil c/\Delta \rceil})$ to construct $s = 2^{O(\Delta)}$ mappings from integers to vectors. For each $i \in [s]$, we use appropriate mapping f_i to construct G_i' with nodes $V_i' = A_i' \cup B_i' \cup C_i'$. For each node $a \in A$, we add d nodes $a_1, ..., a_d$ to A_i' with color a. We construct B_i' and C_i' by adding, for each $b \in B$ and $c \in C$, $2 \cdot 2p$ nodes $b_{j,x}$ and $c_{j,x}$ to B and C respectively, where $j \in [d]$ and $x \in [-p, p]$. We assume that each $v \in V = A \cup B \cup C$ is assigned a unique number in $[3n]$.

We add the following edges to our graph:

A to B : For each edge $(a, b) \in E$, where $(a, b) \in A \times B$, and every $j \in [d]$, place an edge between a_j and $b_{j,x}$, where $x = f_i(w(a, b))[j]$ (in other words, the value of the mapping of $w(a, b)$ at dimension j).

B to C : For each edge $(b,c) \in E$, where $(b,c) \in B \times C$, and every $j \in [d]$: for each $x \in [-p,p]$, if $y = x + f_i(w(b,c))[j] \in [-p,p]$, add an edge between $b_{j,x}$ and $c_{j,y}$.

C to A For each edge $(c,a) \in E$, where $(c,a) \in C \times A$, and every $j \in [d]$, place an edge between $c_{j,x}$ and a_j, where $x = t_i[j] - f_i(w(c,a))[j]$

This constructions gives us $nd + 2 \cdot nd(2p) = O(n^{1+c/\Delta}\Delta)$ nodes, and $2md + 2mdp) = O(mn^{c/\Delta}\Delta)$ edges, and $3n$ colors. We claim that if one of these $2^{O(\Delta)}$ instances of Δ-Matching-Triangles accepts, G contains a triangle of weight 0.

Claim 13. There is a triangle $(a,b,c) \in A \times B \times C$ of weight 0 iff for some $i \in [s]$, there are at least Δ triangles in G'_i with colors (a,b,c).

Proof. For the first direction, take any triangle $(a,b,c) \in A \times B \times C$ in G such that $w(a,b) + w(b,c) + w(c,a) = 0$. By Lemma 11, this means $f_i(w(a,b)) + f_i(w(b,c)) + f_i(w(c,a)) = t_i$ for some $i \in [s]$. By our construction, for each dimension $j \in [d]$, we clearly have triangle $a_j, b_{j_x}, c_{j,y} \in G'_i$, for $x = f_i(w(a,b))[j]$ and $y = x + f_i(w(b,c))[j]$, since we have $y = t_i[j] - f_i(w(c,a))[j]$. Given our assigned coloring, we have $d = \Delta$ triangles of colors (a,b,c) in G_i'.

For the second direction, take any Δ triangles in G'_i for some $i \in [s]$. Each triangle must contain colors a, b, and c, since each of A'_i, B'_i, and C'_i constitute independent sets under our construction. For each $j \in [d]$, a_j can have at most one edge to some unique $b_{j,x}$, and one edge to some unique $c_{j,y}$. This means for each $j \in [d]$, there is at most one triangle with coloring (a,b,c) in G'_i, which exists iff $f_i(w(a,b))[j] + f_i(w(b,c))[j] = t_i[j] - f_i(w(c,a))[j]$. Thus, if there are Δ such triangles, we have vectors $f_i(w(a,b)) + f_i(w(b,c)) + f_i(w(c,a)) = t_i$, and so $w(a,b) + w(b,c) + w(c,a) = 0$. □

Corollary 14. *If there is an algorithm which solves Δ-Matching-Triangles on an n-node graph in time $O(n^{3-\epsilon})$ for an $\epsilon > 0$, $w(1) < \Delta(n) < o(\log n)$, we can solve EW-Triangle in $O(n^{3-\epsilon+o(1)})$ time, and the APSP and 3-SUM conjectures are both false.*

4.5.2 SETH to Matching Triangles

We next give a reduction to Δ-Matching-Triangles under SAT that will complete the proof for a reduction under Conjecture 6. This reduction is similar to those previously mentioned to prove lower bounds conditional on SETH. Unlike the k-SAT to Most-Orthogonal-Vectors reduction shown in Section 4.1, however, this reduction splits the variables into three groups, not two.

Lemma 15. *If Δ-Matching-Triangles can be solved on N-node graphs in time $O(N^{\Delta c})$, then CNF-SAT on n variables and m clauses can be solved in time $O((\Delta 2^{n/3+m/3\Delta})^{c\Delta})$.*

Proof. Given a CNF formula F on n variables and m clauses, we construct graph G as follows. Split the variables up into three groups, U_1, U_2, and U_3, each of size $n/3$, and enumerate over all $N = 2^{n/3}$ partial assignments. Split the clauses up into 3Δ groups, $C_1, ..., C_{3\Delta}$, each of size $m/3\Delta$. For each partial assignment α_i, group of clauses C_{3k+i}, and bit string $s_i \in \{0,1\}^{m/3\Delta}$ (where each bit denotes membership of a given clause in C_{3k+i}), add vertex $v_{\alpha_i,k,s_i} \in V_i$. Assign each partial assignment a different color. Place an edge between nodes

v_{α_i,k,s_i} and v_{α_i+1,k,s_i+1} if α_{i+1} satisfies the subset s_i of C_{3k+i}, and α_i, α_{i+1} satisfy all the clauses of C_{3k+i+1} not included in subset s_{i+1}.

We claim that for all $k \in [\Delta]$, and each triple $\alpha_1, \alpha_2, \alpha_3$ of partial assignments, there is a triangle among some vertices $v_{\alpha_1,k,s_{i_1}}$, $v_{\alpha_2,k,s_{i_2}}$, $v_{\alpha_3,k,s_{i_3}}$ for some i_1, i_2, and i_3 iff they satisfy all clauses in C_{3k+1}, C_3k+2, C_{3k+3}. If there is such a triangle v_{α_1,k,s_1}, v_{α_2,k,s_2}, v_{α_3,k,s_3}, then α_2 must satisfy the subset s_1, and α_1 and α_3 must satisfy the remaining vertices in $C+3k+1$. The same follows for $C+3k+2$ and $C+3k+3$, because there are edges between all three vertices. If α_2 satisfies the subset of clauses in s_1, then v_{α_1,k,s_1} must have an edge to v_{α_2,k,s_j} for all j. The same follows for $i=2$ and $i=3$. Thus all clauses are satisfied by some partial assignment. Since each partial assignment has a different coloring, and there are no edges between $v_{\alpha_i,*,*}$ for any i, it follows that there will be Δ triangles with the same three colors (a,b,c) iff all clauses are satisfied for each $k \in [\Delta]$. This means that, given a $O(N^{c\Delta})$ algorithm for Δ-Matching-Triangles, we can solve CNF-SAT in time $O((\Delta 2^{n/2+m/2\Delta})^{c\Delta})$. $\square$

This, along with the sparsification lemma [8], gives the following corollary.

Corollary 16. *If there is an algorithm to solve Δ-Matching-Triangles on N-node graphs in $O(N^{3-\epsilon})$ time for any $\epsilon > 0$, any $\omega(1) < \Delta(N) < N^{o(1)}$, we can solve k-SAT in $O(2^{n(1-\epsilon/6)})$ time for every $k \geq 3$, and SETH is false.*

This reduction, in combination with that from EW-Triangle proves a lower bound of $n^{3-o(1)}$ for *Delta*-Matching-Triangles conditional on Conjecture 6.

4.5.3 Triangle-Collection to Dynamic Graph Problems

We give a reduction from a restricted version of Triangle-Collection to prove lower bounds on dynamic graph problems. While we do not prove it here, this restricted version can be proven to also have $n^{3-o(1)}$ lower bounds under Conjecture 6 [5].

Definition 13. (Triangle-Collection*) Given an undirected tripartite colored graph G with partitions A, B, C of the following form:

1. A contains $n\Delta$ nodes denoted a_j where $a \in [n]$ and $j \in [\Delta]$ so that a_j is colored with color a.
2. B and C contain $n\Delta p$ nodes each, denoted by $b_{j,x}$ and $c_{j,x}$ where $b, c \in [n]$, $j \in [\Delta]$, and $x \in [p]$ so that $b_{j,x}$ and $c_{j,x}$ are colored b and c, respectively.
3. For each node a_j in A and colors b, c, there is exactly one edge of the form $\{a_j, b_{j,x}\}$ and exactly one edge of the form $\{a_j, c_{j,x}\}$, for some $x, y \in [p]$.
4. A node $b_{j,x}$ in B can only be connected to nodes of the form $c_{j,y}$.

Is it true that for all triples of distinct colors (a,b,c) there is a triangle (x,y,z) in G such that x has color a, y has color b, and z has color c?

Given this reduced version of Triangle-Collection, we have the following lemma on updates and queries of dynamic graphs:

Lemma 17. *Triangle-Collection* can be reduced to $\widetilde{O}(n^2)$ updates and queries of #SSR, #SCC, #SS-Sub-Conn, or Max-Flow on a dynamic graph of $O(n)$ nodes.*

Proof. (#SSR) Given an input graph G to Triangle-Collection*, construct G' as follows: direct all edges between nodes in B and C from B to C; remove all nodes in A; create a source node s and a target node t_c for each color c in C. We can then perform the following operations: first, for every $j \in [\Delta]$, add an edge between $c_{j,x}$ to t_c where $(a_j, c_{j,x})$ is an edge in G; second, we check, for every $j \in [\Delta]$, if for all colors b in B and every edge $(s, b_{j,x})$ such that $(a_j, b_{j,x})$ is an edge in G, we can reach $\Delta + 2n$ from s. After each b, we remove all added edges before moving onto the next color. From the construction we can see that, for any pair $(a_j, b_{j,x})$ in the second stage, we can reach fewer than $\Delta + 2n$ nodes iff there is at least one t_c that we cannot reach from a_j. In other words, we find such a pair iff there is no triangle with this coloring, in which case we can safely reject.

(Max-Flow) The algorithm is identical for that of #SRR, except that we connect each t_c to a single output node t, and weight all other edges n, and check that the maximum flow is equal to the number of nodes t_c.

(#SS-Subgraph-Connectivity) Instead of adding edges as we did in #SSR, we instead turn on the corresponding nodes on any given round, allowing us to again compute whether or not we can reach every t_c from s for a given coloring.

(#Strongly-Connected-Components) Add two new nodes, x_b and x_c, to which we connect each node in B and C respectively bidirectionally. At the first stage, connect each $c_{j,x}$ to t_c instead of x_c; in the second stage, disconnect each $b_{j,x}$ from x_b and instead connect it bidirectionally to s. Thus there is a path from a to every t_c iff, for every coloring, there are precisely 3 strongly connected components: x_b, x_c, and the rest of the graph. □

5 Comparison of Methods

While Backurs and Indyks' original result was quite impressive, the strength of the assumption in SETH is a major weakness. In constrast, the reduction from Branching Programs by Abboud et al. proves identical conditional lower bounds under much weaker assumptions. Arguably, this reduction makes better use of the full power of Edit-Distance and LCS, and thus is a nice follow-up to the work presented in the original paper.

As somewhat of a departure from this, Abboud et al. produce an alternative improvement to the original conditional bounds on SETH. Namely, proving identical lower bounds conditional on *multiple* conjectures. This strengthens any conclusions drawn from reductions under any one conjecture. Furthermore, since they draw on conjectures that span multiple fields, the probability that researchers across all three of these fields have failed to produce even marginally better algorithms for any of these problems makes the idea that at least one conjecture is true all the more credible.

These two tangentially related approaches beg the question of what, then, is the most fruitful strategy to pursue in attempting to prove conditional polynomial lower bounds. The reductions under $\mathcal{C}$-SETH prove very wide-ranging consequences for higher complexity classes should a faster Edit-Distance algorithm be found. However, the conjecture made in the Δ-Matching-Triangles reductions is still fairly weak, even if it does not necessarily prove circuit lower bounds for problems in E^{NP}. Additionally, there is a wide range of further reductions stemming from these original proofs. Thus, while the former has a large impact on the field of complexity theory should faster algorithms for Edit-Distance or LCS be found, the latter provides a good argument that the current upper bounds for a large range of graphs problems may be tight, which has practical consequences in the applications of such problems.

References

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of lcs and other sequence similarity measures. *arXiv preprint arXiv:1501.07053*, 2015.

[2] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for lcs and other sequence similarity measures. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 59–78. IEEE, 2015.

[3] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends or: A polylog shaved is a lower bound made. *arXiv preprint arXiv:1511.06022*, 2015.

[4] Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *Algorithms-ESA 2014*, pages 1–12. Springer, 2014.

[5] Amir Abboud, Virginia Vassilevska Williams, and Huacheng 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*, pages 41–50. ACM, 2015.

[6] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 434–443. IEEE, 2014.

[7] Arturs Backurs and Piotr 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*, pages 51–58. ACM, 2015.

[8] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for sat. In *Computational Complexity, 2006. CCC 2006. Twenty-First Annual IEEE Conference on*, pages 7–pp. IEEE, 2006.

[9] Evgeny Dantsin and Alexander Wolpert. On moderately exponential time for sat. In *Theory and Applications of Satisfiability Testing–SAT 2010*, pages 313–325. Springer, 2010.

[10] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-sat. In *Computational Complexity, 1999. Proceedings. Fourteenth Annual IEEE Conference on*, pages 237–240. IEEE, 1999.

[11] Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 603–610. ACM, 2010.

[12] Virginia Vassilevska and Ryan 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.

[13] Virginia Vassilevska Williams and Ryan 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.