Towards Hardness of Approximation for Polynomial Time Problems*

Amir Abboud[1] **and Arturs Backurs**[2]

1 **Stanford University, Palo Alto, USA**
abboud@cs.stanford.edu
2 **MIT, Cambridge, USA**
backurs@mit.edu

Abstract

Proving hardness of approximation is a major challenge in the field of fine-grained complexity and conditional lower bounds in P. How well can the Longest Common Subsequence (LCS) or the Edit Distance be approximated by an algorithm that runs in near-linear time? In this paper, we make progress towards answering these questions. We introduce a framework that exhibits barriers for truly subquadratic and *deterministic* algorithms with good approximation guarantees. Our framework highlights a novel connection between deterministic *approximation algorithms* for natural problems in P and *circuit lower bounds*.

In particular, we discover a curious connection of the following form: if there exists a $\delta > 0$ such that for all $\varepsilon > 0$ there is a deterministic $(1+\varepsilon)$-approximation algorithm for LCS on two sequences of length n over an alphabet of size $n^{o(1)}$ that runs in $O(n^{2-\delta})$ time, then a certain plausible hypothesis is refuted, and the class E^{NP} does not have non-uniform linear size Valiant Series-Parallel circuits. Thus, designing a "truly subquadratic PTAS" for LCS is as hard as resolving an old open question in complexity theory.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases LCS, Edit Distance, Hardness in P

Digital Object Identifier 10.4230/LIPIcs.ITCS.2017.11

1 Introduction

Canonical examples of problems that are in P due to natural dynamic programming solutions are Edit Distance and Longest Common Subsequence (LCS) [44, 54]. Given two strings x, y of length n, the LCS problem asks for the length of the longest sequence that appears in both x and y as a (not necessarily contiguous) subsequence, and Edit Distance asks to compute the minimum number of operations (insertion, deletion, or substitution) that is required to transform x into y. Despite decades of attempts, it is not known how to speed up the dynamic programming solution beyond the $O(n^2/\log^2 n)$ bound of Masek and Paterson [66] via the "four Russians" technique. Recent research on the exact complexity of polynomial time problems proved that faster algorithms do not exist [19, 2, 32], not even by polylog factors [5], unless SAT can be solved faster than brute force. A natural question arises: How well can we approximate LCS and Edit Distance in truly subquadratic (or near linear) time? And more generally, can we speed up dynamic programming algorithms without paying too much in the optimality of the solution?

* This work was supported in part by an IBM PhD Fellowship, the NSF and the Simons Foundation.

Various generalizations of LCS and Edit Distance, like the Local Alignment problem [75, 8], are fundamental in computational biology and genomics. In such applications, the input size is a few billions, quadratic time algorithms are prohibitive and are rarely run in practice. To analyze the genome, researchers often use various heuristics, like BLAST [13], that run in near linear time but have no optimality guarantees. Despite BLAST's impact, as witnessed by its more than fifty thousand citations, the bioinformatics community is in an everlasting search of "better" algorithms that are able to reveal new phenomena in the massive amounts of biological data that we currently have (see [77, 63, 49]). The theory community ought to provide guidance: is there hope for fast algorithms with strong guarantees? Is there evidence against a $(1+\varepsilon)$-approximation algorithm that runs in near-linear time, even for the more basic LCS and Edit Distance problems?

We say that an algorithm c-approximates the Edit Distance $ED(S,T)$ of two given sequences S,T if it outputs a value x that is $ED(S,T) \leq x \leq c \cdot ED(S,T)$. Since the Edit Distance is at most n, an n-approximation is trivial. A linear time $\sqrt{n}$-approximation follows from the exact algorithm that computes the Edit Distance in time $O(n+d^2)$ where $d = ED(S,T)$ [65]. Subsequently, this approximation factor has been improved to $n^{3/7}$ by Bar-Yossef et al. [21], then to $n^{1/3+o(1)}$ by Batu et al. [22]. Building on the breakthrough embedding of Edit Distance by Ostrovsky and Rabani [69], Andoni and Onak obtained the first near-linear time algorithm with a *subpolynomial* approximation factor of $2^{\tilde{O}(\sqrt{\log n})}$. Most recently, in FOCS'10, Andoni, Krauthgamer, and Onak [15] significantly improved the approximation to polylogarithmic obtaining an algorithm that runs in time $n^{1+\varepsilon}$ and gives $(\log n)^{O(1/\varepsilon)}$ approximation for every fixed $\varepsilon > 0$. There are many works on approximate Edit Distance in various computational models, see e.g. [68, 15, 38] and the references therein.

While LCS and Edit Distance are closely related, they behave quite differently with respect to approximations, and these clever approximation algorithms for Edit Distance are not known to lead to any nontrivial result for LCS. We say that an algorithm c-approximates the LCS of two given sequences S,T if it outputs a value x that is $\frac{LCS(S,T)}{c} \leq x \leq LCS(S,T)$. A cute observation shows that the LCS of binary sequences can be approximated to a factor of 2 in linear time: the longest common subsequence that is all-zero or all-one is at least half from optimal. Note that a 2 approximation for Edit Distance on binary sequences would be a breakthrough. In general, for an alphabet of size $|\Sigma| = s$, it is easy to get an s-approximation for the LCS and it is a longstanding open question to design an $(s-\delta)$-approximation in near-linear time or even strongly subquadratic time for any constant integer $s \geq 2$ and constant $\delta > 0$. Even though many ideas and heuristics for LCS were designed [43, 27, 48, 45] (see also [68, 28] for surveys), none has proven sufficient to compute an $(s-\delta)$-approximation in strongly subquadratic time. A general tool for speeding up dynamic programming algorithms through a controlled relaxation of the optimality constraint could have great impact for algorithm design. Recently, encouraging positive results along these lines were obtained by Saha [72, 73] for problems related to parsing context-free languages. However, we are still far from understanding, more generally, when and how such speedups are possible.

Proving lower bounds for problems in P, under popular conjectures like the Strong Exponential Time Hypothesis (SETH) [57, 35], is a recent and very active line of work [2, 4, 6, 7, 8, 9, 14, 19, 32, 33, 31, 30, 42, 70, 71, 79, 1, 18, 20, 47, 41, 40]. The known results for LCS and Edit Distance [19, 2, 32, 5] do not imply any non-trivial hardness of approximation, i.e. they only rule out (roughly) $(1+1/n)$-approximations in subquadratic time. Achieving strong hardness of approximations results is often highlighted as an important open question for this line of research, and the general sense of the community is that new ideas that deviate significantly from current techniques might be required.

Our Work

In this paper, we make progress towards the important goal of proving inapproximability results for such fundamental problems in P. We introduce a framework that exhibits barriers for subquadratic and *deterministic* algorithms with good approximation guarantees. Admittedly, the "lower bounds" we obtain for problems like LCS are quite weak and are still far from the upper bounds. Still, they are *much* higher than what we knew before: e.g. instead of the trivial $(1 + 1/n)$-approximation hardness, we can show evidence against $(1 + 1/\text{poly} \log n)$ or even $(1 + o(1))$ approximations. Perhaps more interesting than the statements is the framework itself, which highlights a novel connection between deterministic *approximation algorithms* for natural problems in P and *circuit lower bounds*.

We prove a curious connection of the following form: if there is a truly subquadratic deterministic $(1 + \varepsilon)$-approximation algorithm for LCS on two sequences of length n over an alphabet of size $n^{o(1)}$, then the complexity class E^{NP} does not have non-uniform linear size series parallel circuits[1]. This consequence (explained in more detail below) is widely believed to be true. However, proving it unconditionally would be a breakthrough in complexity theory and the study of non-uniform circuit lower bounds. As stated, this is merely a "difficulty" or a "no-pass" result for LCS, not "hardness". It only shows a "circuit lower bounds" barrier for designing a fast $(1 + o(1))$-approximation algorithm for LCS: it is at least as difficult as resolving a longstanding (and considered to be difficult) open question in circuit complexity. However, we prove a stronger result (Theorem 5 below), which we think should be regarded as a "hardness" result as well, giving evidence that such algorithms for LCS might not exist.

We contribute to the growing body of surprising connections between algorithm design and circuit lower bounds (see the recent survey [85]) [62, 56, 83, 86, 61, 50, 37, 16, 59, 60, 64]. A notable tight connection between faster algorithms for Circuit-SAT and circuit lower bounds was shown by Williams [83, 86]: faster-than-trivial Circuit-SAT algorithms for many circuit classes $\mathcal{C}$ imply interesting new lower bounds against that class. For example, via this connection, Jahanjou, Miles, and Viola [60] show that refuting SETH leads to proving the same lower bound against series-parallel circuits stated above. Abboud et al. [5] go a step further and show that slightly faster algorithms for natural and well-studied problems in P (as opposed to Circuit-SAT) are enough to prove lower bounds against large classes like non-uniform NC. A related intriguing connection is between *derandomization* (of algorithms and circuits) and circuit lower bounds [56, 61, 84, 74, 26, 12]. A derandomization algorithm for a circuit class $\mathcal{C}$ is a deterministic algorithm that is able to distinguish, given a circuit from $\mathcal{C}$, whether it is unsatisfiable (zero satisfying assignments) or "very satisfiable" (at least $2^n \cdot (1 - o(1))$ satisfying assignments). Note that a derandomization algorithm can be obtained from an algorithm that approximates the number of satisfying assignments to circuits from $\mathcal{C}$ (known as CAPP - Circuit Acceptance Probability Problem). Combining the framework of Williams [83] with a "Succinct PCP" [67, 26] shows that to prove a lower bound against a class $\mathcal{C}$ it is enough to obtain a nontrivial *derandomization* algorithm for a class $\mathcal{C}'$ (that could be slightly larger than $\mathcal{C}$) [83, 74, 26]. Our work connects circuit lower bounds, via circuit derandomization tasks, to designing approximation algorithms for natural optimization problems in P like LCS, as opposed to CAPP or algebraic problems like polynomial identity testing (Williams [87] recently showed that derandomizing a quadratic time algorithm for a variant of this problem implies interesting circuit lower bounds).

[1] The class E^{NP} or $\mathsf{TIME}[2^{O(n)}]^{\mathsf{NP}}$ is the class of problems solvable in exponential time with access to an NP oracle.

1.1 Our Results

We will now give a more detailed overview of our results, and then in Section 3 we present the technical details of our framework. Section 1.3 will discuss how our approach could lead to further hardness of approximation results for problems in P. The complete proofs are given in the subsequent sections.

The Gap Block Disjointness Hypothesis

Our result for LCS will be based on the presumed difficulty of solving the following Gap Block Disjointness (GBD) problem in subquadratic time.

▶ **Definition 1** (Gap Block Disjointness)**.** Given two lists of Boolean matrices $A, B \subseteq \{0,1\}^{K \times D}$ of size $|A| = |B| = N$, we say that a pair $A_i \in A, B_j \in B$ is a "good pair" if there exists a $k \in [K]$ such that the rows $A_i(k, \cdot)$ and $B_j(k, \cdot)$ are disjoint, i.e.

$$\vee_{k \in [K]}(\wedge_{h \in [D]}(\neg A_i(k,h) \vee \neg B_j(k,h))) = 0.$$

The Gap Block Disjointness problem is to decide whether we are in case 1 or in case 2 (and if we are in neither, the output can be arbitrary):

1. (zero "good" pairs) none of the pairs $A_i \in A, B_j \in B$ are good.

$$\Pr_{i,j \in [N]}\left[\ \vee_{k \in [K]}(\wedge_{h \in [D]}(\neg A_i(k,h) \vee \neg B_j(k,h))) = 0\ \right] = 0$$

2. (many "good" pairs) at least $N^2 \cdot (1 - 1/\log_2^{10} N)$ pairs $A_i \in A, B_j \in B$ are good.

$$\Pr_{i,j \in [N]}\left[\ \vee_{k \in [K]}(\wedge_{h \in [D]}(\neg A_i(k,h) \vee \neg B_j(k,h))) = 1\ \right] \geq (1 - 1/\log_2^{10} N)$$

A trivial algorithm solves this problem in quadratic time, by going over all pairs of matrices, but can we do better? Note that if we ask whether at least one "good pair" exists (without the above gap-promise) then the problem requires $N^{2-o(1)}$ under SETH (which is conjectured to hold even for randomized algorithms), even when $K = 1$ and $D = \Omega(\log N)$, since this is the Orthogonal Vectors problem [82, 10, 39]. We introduce the hypothesis that this *gap* version, with $D = \Omega(\log N)$ and $K = N^{o(1)}$, cannot be solved by a deterministic algorithm in truly subquadratic time.

▶ **Hypothesis 2.** *There is no $\varepsilon > 0$ and $\alpha > 0$ such that for all constant d we can solve the Gap Block Disjointness problem on binary matrices in $n^\alpha \times d \log n$ in deterministic $O(n^{2-\varepsilon})$ time.*

Interestingly, unlike all previous hardness conjectures in the "Hardness in P" research [52, 70, 79, 6, 9, 55, 3, 11], ours does not remain plausible when faced against randomized algorithms. A near-linear time randomized algorithm that samples a few pairs can easily solve this problem, with high probability. But can a *deterministic* algorithm do anything clever enough to solve the problem in truly subquadratic time? Such an algorithm is not known, and in fact, Lemma 3 below suggests that it would be a breakthrough.

Series-parallel circuits [78, 34, 81, 46] (or VSP circuits) are special kind of circuits that can be obtained by combining circuits either in series or in parallel (defined formally in Section 2). In 1977, Valiant introduced these circuits and argued that most known computer programs fit under this restriction. His hope was that understanding these circuits would be easier than the general case. Four decades later, we still do not know how to resolve basic challenges proposed in his paper, like showing an explicit function that does not have *linear*

size series parallel circuits. It is still conceivable that the large class E^{NP} can be computed by such circuits, and proving otherwise would be a major achievement. Our first lemma states that refuting Hypothesis 2 is at least as difficult as showing these results.

▶ **Lemma 3.** *If Hypothesis 2 is false, then the class* E^{NP} *does not have non-uniform linear size* VSP *circuits.*

A reader familiar with previous SETH lower bounds might wonder why we need this GBD problem, as opposed to simply considering the $K = 1$ case, i.e. the gap version of Orthogonal Vectors. Without going into the details, we remark that, as far as we can show, a faster deterministic algorithm for that case would not imply any new circuit lower bound. Intuitively, this is because the $K = 1$ case can only encode CNF formulas, which is an extremely weak computational model, for which the corresponding circuit lower bounds are easy to prove unconditionally.

We stress that this circuit lower bound consequence is only meant to show that the hypothesis is hard to *refute*. As evidence that the hypothesis is *plausible*, we remark that none of the current (e.g. [39, 53]) or conjectured-to-exist derandomization techniques (e.g. if $\mathsf{P} = \mathsf{BPP}$) are enough to refute it. While a common belief is that randomized algorithms cannot outperform deterministic ones by more than a polynomial factor, it is plausible that randomization can give, say, a linear $\Omega(n)$ speedup.

Reduction to Approximate LCS

The simplicity of the GBD problem makes Hypothesis 2 an appealing starting point for proving barriers. Our main technical lemma shows how GBD can be reduced to LCS while creating a multiplicative gap, giving the first nontrivial hardness of approximation result for LCS.

▶ **Lemma 4.** *If for some* $\delta > 0$*, there is a deterministic algorithm that can approximate the LCS of two given sequences of length* n *over an alphabet of size* $n^{o(1)}$ *to within a* $(1 + \varepsilon)$ *factor, for all* $\varepsilon > 0$*, in* $O(n^{2-\delta})$ *time, then Hypothesis 2 is false.*

Together, these two lemmas imply our main theorem:

▶ **Theorem 5.** *If for some* $\delta > 0$ *there is a deterministic* $(1 + o(1))$*-approximation algorithm for LCS on two sequences of length* n *over an alphabet of size* $n^{o(1)}$ *in* $O(n^{2-\delta})$ *time, then Hypothesis 2 is false and the class* E^{NP} *does not have non-uniform linear size* VSP *circuits.*

We remark that our hardness for approximate LCS immediately transfers to nontrivial results for other problems. For example, we get that the RNA Folding problem which is central in computational biology [2, 17, 51, 76, 80] cannot be approximated to within in a $(1 + o(1))$ factor in truly subquadratic time.

A simple application of our framework, gives a weaker lower bound for Edit Distance and LCS on *binary* sequences. In Section 4.1, we show that a deterministic $(1 + 1/\mathrm{poly}\log n)$-approximation for these problems in truly subquadratic time implies that E^{NP} does not have log-depth (NC^1) circuits.

It is likely that more efficient reductions from GBD to LCS (and Edit Distance) can be devised, and if certain gadgets in our proof can be implemented more efficiently, a *tight* conditional lower bound against $(2 - \delta)$-approximations for LCS on binary sequences could follow. Moreover, better gadgetry would be able to boost the consequence from a VSP circuit lower bound through GBD to a stronger circuit lower bound as in [5]. On a different

note, we believe that a further tightening of the connections between nontrivial circuit derandomization and approximation algorithm for extensively studied problems like LCS and Edit Distance could be a promising direction for *proving* new circuit lower bounds. Perhaps, via stronger connections, one would be able to use highly involved algorithms like Andoni et al.'s approximation for Edit Distance [15] to prove new breakthroughs in complexity theory.

1.2 Technical Overview

To motivate our framework, we give a short exposition of the known constructions for hardness of sequence alignment problems, and why they fail to give any nontrivial consequences of approximation algorithms.

For concreteness, consider the reductions from CNF-SAT to LCS used to prove that LCS requires $n^{2-o(1)}$ time under SETH [2, 32]. In these reductions, we take a CNF formula on n variables and m clauses, say $m = O(n)$, and produce two sequence of length $N \cdot \operatorname{poly}\log N$ where $N = 2^{n/2}$, so that the LCS of the two sequences is large iff the formula is satisfiable. Each sequence is composed of $O(N)$ segments of length $O(\log N)$ called *assignment gadgets*, representing all $2^{n/2}$ partial assignments to half of the variables (each half of the variables is represented in the gadgets in one of the two sequences). When two assignment gadgets are "matched" in an LCS alignment, the contribution to the total score is X_{sat} if the two corresponding partial assignments make up a satisfying assignment to our CNF formula and X_{unsat} otherwise, where $X_{sat} > X_{unsat}$. Due to the way these gadgets are composed, the optimal LCS will be achieved by matching roughly $N = 2^{n/2}$ pairs and therefore gaining a score that is at most $N \cdot X_{unsat}$ if the formula is unsatisfiable, and at least $(N-1) \cdot X_{unsat} + X_{sat}$ if it is satisfiable. Now, notice that X_{sat} cannot be more than $O(\log N)$, since it is upper bounded by the length of the assignment gadgets, while X_{unsat} is at least 1, since otherwise the gadgets do not encode enough information. This implies that the multiplicative gap between the two cases is no more than $(1 + 1/N)$.

A natural attempt to increase this gap is by using a PCP theorem on the initial CNF formula before reducing it to LCS. For instance, this approach has been used in many ETH based lower bounds [29] [2]. Even if an ultra efficient PCP theorem existed, where the number of variables remains $(1 + o(1)) \cdot n$ and the gap increases arbitrarily, this approach does not give any interesting hardness for approximate LCS: The PCP will only affect the gap between X_{sat} and X_{unsat}, which only affects the low order terms in the total score: it is always upper bounded by the length of the gadgets $X_{sat}/X_{unsat} = N^{o(1)}$, and so the multiplicative gap remains $(1 + N^{o(1)}/N)$. The problem with this approach is that the dominant factor is *not* how many clauses our best assignment satisfies, but it is how many *assignments* satisfy all clauses. Standard PCP approaches make unsatisfying assignments less satisfying, but they do not affect the *number of satisfying assignments*.

This leads to another natural attempt: can we find a different kind of gap amplification result that reduces a formula f to another formula f' so that f' has *many* satisfying assignments iff f is satisfiable? To get interesting hardness results for LCS, we will need the gap between the two cases to be quite large, e.g. zero satisfying assignments vs. at least $2^n/10$ satisfying assignments. We would like to do this while keeping the number of variables $(1 + o(1)) \cdot n$. Unfortunately, this kind of gap amplification for CNF's and circuits is unlikely as it would lead to a randomized polynomial time algorithm for CNF-SAT ($\mathsf{BPP} = \mathsf{NP}$): perform the reduction then sample $O(n)$ assignments and check if they satisfy f'.

[2] ETH states that 3-SAT cannot be solved in $2^{o(n)}$ time. Obtaining tight lower bounds for problems in P under ETH is a major open question.

While it is easy for a randomized algorithm to distinguish between unsatisfiable formulas (or circuits) and almost-completely satisfiable ones, the main observation at the base of our framework is that this task might not be so easy for *deterministic* algorithms. Given a circuit on n variables that has $2^n - 2^n/n^{10}$ satisfying assignment, how can a deterministic algorithm find one of its satisfying assignments? If the algorithm treats the circuit as a black-box and blindly queries it with assignments until it outputs 1, the runtime will not be $O(2^{(1-\varepsilon)n})$. Otherwise, the algorithm could try to analyze the circuit and understand its satisfiability properties in order to achieve faster deterministic runtime. The central insight in the connections between algorithms and lower bounds [85] is that our ability to design algorithms for analyzing circuits (from a certain class $\mathcal{C}$) is closely related to our ability to show limitations of circuits (lower bounds against the class $\mathcal{C}$). In fact, there are formal and quite tight connections showing that a faster-than-trivial deterministic algorithm for the "circuit-derandomization" problem above, on certain classes of circuits, implies new circuit lower bounds.

Our framework takes this route in order to get evidence for difficulty and hardness of designing approximation algorithms. Depending on the target problem (in P) for which we seek a "lower bound", one might want to start from a different derandomization problem concerning a different class of circuits so that it embeds as efficiently as possible into the problem. In this paper, we instantiate the framework with the class of linear size series parallel circuits and prove that they embed nicely into approximate LCS (via the GBD problem).

We remark that all our results for consequences of *deterministic* algorithms remain valid for (appropriately defined) *co-nondeterministic* algorithms (see [36] for interesting results on this notion). It is difficult to approximate LCS even with nondeterminism.

Derandomization implies Circuit Lower Bounds

The connection between derandomizing circuits and lower bounds originates in the works of Impagliazzo, Kabanets, and Wigderson [56] and has been optimized significantly by the work of Williams [83], Santhanam and Williams [74], and more recently by Ben-Sasson and Viola [26]. These connections rely on "Succinct PCP" theorems [24, 67, 25, 23, 26], and the recent optimized construction of Ben-Sasson and Viola [26] is crucial to our main result.

Our starting point is the following theorem (Theorem 1.4 in [26]), which we will state more formally later in the paper: Let F_n be a set of functions from $\{0,1\}^n$ to $\{0,1\}$ that satisfies some minor requirements (e.g. functions that can be computed by linear size circuits from some class $\mathcal{C}$). If the acceptance probability of a function of the form

- AND of fan-in $n^{O(1)}$
- of OR's of fan-in 3
- of functions from $F_{n+O(\log n)}$

can be distinguished from being $= 1$ or $\leq 1/n^{10}$ in $2^n/n^{\omega(1)}$ deterministic time, then there is a function f in E^{NP} on n variables such that $f \notin F_n$ (and therefore cannot be computed by linear size circuits from $\mathcal{C}$, and E^{NP} is not contained in non-uniform $\mathcal{C}$).

The optimization of the Succinct PCP by Ben-Sasson and Viola makes the overhead in this connection quite small: we only need two additional levels to the original circuit class (the AND and OR), one of which has fan-in 3. When instantiating this theorem with linear size VSP circuits, this minor overhead allows us to still obtain a simple enough class of circuits that allows for an efficient reduction to our GBD problem (and then to approximate LCS).

Next, we show a reduction from the derandomization task above of AND-OR-VSP circuits to the GBD problem. This reduction is obtained via a series of transformations to the circuit, so that we end up with an OR-AND-OR circuit of the following form, for some constants ℓ, c, μ, which embeds nicely into GBD:

- OR of fan-in $n^{O(1)} \cdot 2^{n/\ell} \cdot 2^{\varepsilon n}$,
- of AND of fan-in $f(\varepsilon, k) \cdot n$ where $k = 2^{2^{\mu c \ell}}$,
- of OR of fan-in k of literals.

To get this form, we use the classical depth reduction of Valiant [78] which is especially powerful for VSP circuits, as well as the sparsification lemma for CNF formulas [57, 58]. The details of Valiant's depth reduction theorem were clarified by Calabro [34] and Viola [81] (cf. Cygan et al. [46]). To reduce the derandomization problem of such AND-OR-AND circuits to GBD we follow the split-and-list technique, similarly to the reduction from CNF-SAT to Orthogonal Vectors [82]. Choosing all the parameters carefully, we get Lemma 3.

Reducing to Approximate LCS

The reduction from GBD to approximate LCS has two main components: an *inner construction*, in which we encode each of the matrices separately into (short) "matrix" or "inner" gadgets, and an *outer construction* that combines the inner gadgets into two final (long) sequences. This outer and inner outline is not different from previous reductions to LCS and Edit Distance, except that now will we have to make sure that our constructions preserve multiplicative gaps. Getting a gap in either of these constructions was beyond previous techniques, and our work contributes to both: The gap in the outer construction will follow, for the most part, from our starting point (GBD and the derandomization problems as opposed to SAT). For the inner construction, however, we need to design new gadgetry that could be of interest even beyond our "difficulty via derandomization" framework. We explain the main ideas below.

The inner construction: In the inner construction we map each one of our input matrices $A_i \in A$ into a sequence a_i and each of our $B_j \in B$ into a sequence b_j so that there is a value X_{bad} and a constant $\varepsilon > 0$ for which: $LCS(a_i, b_j) \leq X_{bad}$ if A_i, B_j is not a good pair, while if A_i, B_j is a good pair then the LCS is much larger $LCS(a_i, b_j) \geq (1 + \varepsilon) \cdot X_{bad}$.

Constructions from previous work [2, 32, 5] easily give such "matrix" gadgets, except the gap between the two cases would be too small: X_{bad} vs $X_{bad} \cdot (1 + 1/n^{\alpha})$. Instead, we introduce some new ideas that lead to a $(1 + \varepsilon)$ gap at the cost of less efficiency in terms of the length of the gadgets and the alphabet size.

We will first construct "disjointness" gadgets that encode each row of each of our matrices. For this discussion, fix a pair of matrices $A_i \in A, B_j \in B$. We will first encode each of A_i's rows $A_i(k, \cdot)$ with a sequence $a_{i,k}$ and each of B_j's rows $B_j(k, \cdot)$ with a sequence $b_{j,k}$ (these are the disjointness gadgets). Our goal will be to have $LCS(a_{i,k}, b_{j,k})$ be $X_{intersect}$ if the two rows intersect, and at least $(1+\varepsilon) \cdot X_{intersect}$ if the rows are disjoint. To achieve this, we use a new encoding that is specifically tailored towards LCS on large alphabets. The idea of our encoding is to chop each of our rows of length $d \log n$ into ℓ pieces of length $d \log n / \ell$ each, and then think of each piece as a separate letter from an alphabet of size $2^{d \log n/\ell} = n^{o(1)}$. Then, we let $a_{i,k}$ be the concatenation of ℓ such symbols, corresponding to the ℓ pieces that appear in the row $A_i(k, \cdot)$. Meanwhile, $b_{j,k}$ will be defined differently: for each piece x, we will enumerate all possible letters σ that correspond to pieces y that are disjoint from x, and write them in a dedicated segment in $b_{j,k}$. By doing this, the LCS of $a_{i,k}$ and $b_{j,k}$ will be exactly ℓ if the two rows are disjoint (since all pieces will be disjoint and contribute a letter

to the LCS), while if the rows intersect the LCS will be at most $\ell - 1$. Since we can afford to pick ℓ to be a (large enough) constant, we obtain a multiplication gap of $(1 + \varepsilon)$ where $\varepsilon = 1/(\ell - 1)$ is a constant.

Next, we need to combine these "disjointness" gadgets into "matrix gadgets" with an OR: we want the score to be large iff *there exists* a $k \in [K]$ for which the rows are disjoint. Previously known OR gadgets are not sufficient: the total score would contain a sum over all $k \in [K]$ of the score of the k^{th} disjointness gadget, which would decrease the gap back to $(1 + 1/K) = (1 + 1/n^{\alpha})$. To overcome this, we use a different OR gadget that heavily abuses the alphabet in order to keep the multiplicative gap *unchanged*. The idea is simple: let us designate a separate alphabet Σ_k for the k^{th} disjointness gadget $a_{i,k}$ or $b_{j,k}$ (representing the k^{th} row of A_i or B_j), so that for $k \neq k'$ the alphabets are disjoint $\Sigma_k \cap \Sigma_{k'} = \emptyset$. And now our matrix gadgets, which are an OR of our disjointness gadgets, are defined as follows:

$$a_i := a_{i,1}\ a_{i,2}\ \cdots\ a_{i,k}$$

$$b_j := a_{j,k}\ b_{j,k-1}\ \cdots\ b_{j,1}$$

The extremely useful property of this construction is that the LCS is the *maximum* over k of the LCS of $a_{i,k}$ and $b_{j,k}$, as opposed to any expression with a summation over all k. To see this, first note that only letters from gadgets with the same index k can be matched, and now imagine we pick at least one matching for some k, say for $k = 1$ so that we matched some letter between $a_{i,1}$ and $b_{j,1}$, and notice that now we can no longer match any letters from gadgets with a different index $k' \neq k$ without creating a crossing. Therefore, we get that the score of these matrix gadgets is *exactly the same* as the score of the best disjointness gadget across all rows $k \in [K]$, and so if A_i, B_j is a bad pair the score cannot be more than $X_{bad} = X_{intersect}$, while if the pair is good the score is $X_{good} = X_{disjoint} \geq (1+\varepsilon) \cdot X_{intersect}$, where ε is the same constant defined above.

For the outer construction, we use a similar "alignment gadget" to the one used in previous works [2, 32], but we need to analyze it more carefully in order to argue that it generates a gap when working with a gap problem like GBD. We show that if we are in case 2, then there is a matching that contains a large number of pairs A_i, B_j each contributing $(1 + \varepsilon) \cdot X_{bad}$, while in case 1 all pairs in all optimal matchings will contribute only X_{bad}. While previous work used padding of size that is linear in the size of the inner gadgets, we will have to work with much smaller paddings of size that is linear only in the LCS between the inner gadgets. By a careful choice of the parameters we show that this is a $(1 + \varepsilon')$ gap, for some $\varepsilon' > 0$.

1.3 Discussion

Fundamentally, our approach is based on the following intuition. If there is a search problem that we do not expect any (deterministic or randomized) algorithm to be able to solve much faster than brute force, like the problem of finding a pair of vectors that satisfy some function (as in GBD and Orthogonal Vectors), then we might expect the *gap* version of the problem to be hard for deterministic algorithms: maybe we cannot even distinguish the case in which no "good solutions" exist in the search space from the case that almost any solution is "good".

In the context of circuit lower bound consequences from circuit analysis algorithms, this intuition is more or less formal: most lower bound consequences we can get from SAT algorithm follow also from such a distinguisher.

Does the same hold with respect to the other conjectures used in "Hardness in P" research? Consider the 3-SUM problem which asks if a set of n numbers contains three that sum to zero, and is conjectured to require $n^{2-o(1)}$ time. If we believe the 3-SUM conjecture, should

we also believe that we cannot deterministically distinguish an input with many triples that sum to zero from an instance with few? Would this "Gap 3-SUM Conjecture" have interesting consequences? We believe that this is an intriguing avenue for future research and expect it to be fruitful, either in terms of conditional lower bounds, or in terms of a better understanding of our conjectured-to-be-hard problems.

2 Valiant series-parallel circuits

▶ **Definition 6** (Valiant series-parallel graphs [78, 34, 81, 46]). A *multidag* $G = (V, E)$ is a directed acyclic multigraph. Let input(G) be the set of vertices of G with in-degree 0. Let output(G) be the set of vertices of G with out-degree 0. We say that the multidag G is a Valiant series parallel (VSP) graph if there exists a *labelling* $l : V \to \mathbb{Z}$ of G with the following properties:

- For all directed edges $(u, v) \in E$ we have that $l(u) < l(v)$.
- There exists an integer $d \in \mathbb{Z}$ such that for all $v \in$ input(G), $l(v) = d$. The definition from [34] asks that $d = 0$. It is not hard to verify that our definition is equivalent to theirs.
- There exists an integer $d' \in \mathbb{Z}$ such that for all $v \in$ output(G), $l(v) = d'$.
- There is *no* pair of directed edges $(u, v), (u', v') \in E$ such that the inequality $l(u) < l(u') < l(v) < l(v')$ holds.

▶ **Definition 7** (Valiant series-parallel circuits [78, 34, 81, 46]). A circuit is a Valiant series-parallel circuit if the underlying multidag is a VSP graph and the fan-in of every gate is at most 2.

▶ **Definition 8** (Size of a circuit). The size of a circuit on n input variables is equal to the number of gates in it. We do not count the $n + 2$ input nodes, i.e. the input variables and the two constant values 0 and 1 (which are assumed to be given as the last two input nodes to a circuit).

▶ **Definition 9** (VSP_c). We define class VSP_c to be the set of languages recognizable by VSP circuits of size at most $\leq cn$ where n is the number of input variables. The set of allowed gates is the set of *all* gates of fan-in at most 2.

Below we show properties of the class VSP_c that we will use later in the paper.
We need the following definition from [26].

▶ **Definition 10** ([26]). Let F_n be a set of functions from $\{0,1\}^n$ to $\{0,1\}$. We say that F_n is *efficiently closed under projections* if functions in F_n have a poly(n)-size description and given (the description of) a function $f \in F_n$, indexes $i, j \leq n$, and a bit b, we can compute in time poly(n) the functions $\neg f$, $f(x_1, \ldots, x_{i-1}, b \text{ XOR } x_j, x_{i+1}, \ldots, x_n)$, and $f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$, all of which are in F_n.

▶ **Lemma 11.** *The class* VSP_c *is efficiently closed under projections for any constant* $c > 1$.

Proof. From Definition 9 it follows that the class VSP_c has poly(n) description: the circuit itself. Consider a function f on n input variables from VSP_c that has a VSP circuit of size at most $\leq cn$. We show that the three functions from the statement of Definition 10 can be computed in poly(n) time and that all of them are in VSP_c.

Function $\neg f$

Consider the output. If it is one of the inputs, we add a NOT gate and remove all the other gates. If the output is not one of the inputs, it must be some gate g. We replace it with gate $\neg g$. Since allow *all* gates of fan-in at most 2, we can do this. The number of gates did not increase and the function $\neg f$ is now in VSP_c. Clearly, the transformation can be done in poly(n) time.

Function $f(x_1, \ldots, x_{i-1}, b \text{ XOR } x_j, x_{i+1}, \ldots, x_n)$

If $b = 0$, we rewire all gates that used input x_i to use input x_j. If $b = 1$, we rewire all gates to use NOT x_j. Since we have all gates of fan in at most 2, we don't need to introduce the NOT gate. Instead, we replace the gate by another gate that negates the corresponding input. Similarly as before, the transformation can be done in poly(n) time and we did not increase the number of gates. Thus, the resulting function is in VSP_c.

Function $f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$

The transformation is similar as in the previous case. Instead of rewiring to x_j, we rewire to the constant function 0 which is among the inputs. ◀

▶ **Lemma 12.** *Let $f_1, f_2, f_3 \in \mathsf{VSP}_c$ be three functions on n variables from the class VSP_c. Then the function*

$$f := \neg(f_1 \ OR \ f_2 \ OR \ f_3)$$

on the same n variables belongs to VSP_{4c} if $c \geq 4$ and $n \geq 10$.

Proof. For every $i = 1, 2, 3$, let C_i be the VSP circuit of size $\leq cn$ corresponding to the function f_i, and let G_i be the underlying VSP multidag of C_i. Let the multidag $G := P(G_1, G_2, G_3)$ be the *disjoint* union of the underlying VSP multidags G_1, G_2, G_3, and let $\text{input}(G_i) = \{u_i^1, \ldots, u_i^n, u_i^{n+1}, u_i^{n+2}\}$ be the $n + 2$ input nodes for C_i, $i = 1, 2, 3$ (see Definition 8), where the first n nodes $u_i^1, \ldots, u_i^n$ correspond to the n input variables, and u_i^{n+1} and u_i^{n+2} correspond to the two constant functions 0 and 1, respectively. We have that $\text{input}(G) = \text{input}(G_1) \cup \text{input}(G_2) \cup \text{input}(G_3)$. Moreover, since $|\text{input}(G_i)| = n + 2$, $|\text{input}(G)| = 3n + 6$. Each C_i has only one output gate. Thus, $\text{output}(G_i) = \{o_i\}$ for some node o_i. Therefore, $|\text{output}(G)| = 3$.

A disjoint union of two VSP multidags is a multidag (see the proof of Lemma 3 in [34]). Therefore, the multidag G is a VSP multidag. Let, then, l be the labeling of G according to Definition 6 of VSP graphs. We construct a circuit C for the function f as follows. First, we let C be the disjoint union of C_1, C_2, C_3 (each C_i has its own $n + 2$ input nodes). Therefore, the underlying graph of C is G. Next, whenever we add a node or an edge to G, we do the same for C, and the other way around. As the circuits C_1, C_2, C_3 do not share their inputs, we add $n + 2$ input nodes $u^1, \ldots, u^n, u^{n+1}, u^{n+2}$ to C (and to G). The first n input nodes $u^1, \ldots, u^n$ correspond to the n input variables, and the 2 input nodes u^{n+1} and u^{n+2} correspond to the two constant function 0 and 1, respectively. We connect the n input nodes $u^1, \ldots, u^n$ in pairs to the first n input nodes of C_i (for every $i = 1, 2, 3$). That is, for every $j = 1, \ldots, n$ and $i = 1, 2, 3$, we connect u^j to u_i^j. In addition, for every $j = n + 1, n + 2$ and $i = 1, 2, 3$, we connect u^j to u_i^j.

For every newly added input node u^j, $j = 1, \ldots, n+2$, we update the labeling: $l(u^j) = d-1$. As a result, the multidag G has $\text{input}(G) = \{u^1, \ldots, u^n, u^{n+1}, u^{n+2}\}$ and all weights of these

nodes are equal to $d-1$. It remain to verify the fourth property of VSP graphs. Since for every u^j, if (u^j, v) is an edge in G, then $l(v) = d$, the fourth property also holds. Thus G is VSP graph.

Now we have three functions f_1, f_2, f_3 on the same set of $n+2$ inputs. To get the function $f = \neg(f_1 \text{ OR } f_2 \text{ OR } f_3)$, we add two more gates u_1, u_2 to the circuit C. We set the labeling: $l(u_1) = d' + 1$ and $l(u_2) = d' + 2$. u_1 is an OR gate, and it computes the OR of o_1 and o_2 (the outputs of the functions f_1 and f_2). The gate u_2 is a $\neg$OR gate, and it computes the negation of the OR of u_1 (the OR of f_1 and f_2) and o_3 (the output of the function f_3). Since all gates of fan-in at most two are allowed, we can implement a $\neg$OR gate. We can check that C computes f (the negation of the OR of f_1, f_2, f_3). The size of the circuit C is at most $3cn + |\text{input}(G)| + 2 = 3cn + 3n + 8 \leq 4cn$ as required. It is not hard to verify that the resulting labeling of u_1, u_2 and the rest of the multigraph G satisfies the properties from Definition 6. Thus, we conclude that the resulting underlying multidag G is a VSP graph and that C is a VSP_{4cn} circuit. ◀

3 VSP Circuits and Block Disjointness

Circuit Lower Bounds from Derandomization

The connection between derandomizing circuits and lower bounds originates in the works of Impagliazzo, Kabanets, and Wigderson [56] and has been optimized significantly by the work of Williams [83], Santhanam and Williams [74], and more recently by Ben-Sasson and Viola [26]. These connections rely on "Succinct PCP" theorems [67, 26], and the recent optimized construction of Ben-Sasson and Viola [26] is crucial to our main result. Our starting point is the following theorem.

▶ **Theorem 13** (Theorem 1.4 in [26]). *Let F_n be a set of function from $\{0,1\}^n$ to $\{0,1\}$ that are efficiently closed under projections (see Definition 10).*

If the acceptance probability of a function of the form

- *AND of fan-in $n^{O(1)}$ of*
- *OR's of fan-in 3 of*
- *functions from $F_{n+O(\log n)}$*

can be distinguished from being $= 1$ or $\leq 1/n^{10}$ in TIME$(2^n/n^{\omega(1)})$, then there is a function f in E^{NP} on n variables such that $f \notin F_n$.

The optimization of the Succinct PCP by Ben-Sasson and Viola makes the overhead in this connection quite small: only two additional levels to the circuit, one of which has fan-in 3. Next, we instantiate this theorem with VSP circuits and then do simple tricks to the circuits in order to simplify the derandomization task as much as possible.

▶ **Lemma 14.** *To prove that E^{NP} does not have non-uniform Valiant series parallel* (VSP) *circuits of size cn on n input variables, it is enough to show a deterministic algorithm for the following Derand-VSP problem that runs in $2^n/n^{\omega(1)}$ time. Given a circuit over n input variables of the form:*

- *OR of fan-in $n^{O(1)}$ of*
- *negations of OR's of fan-in 3 of*
- VSP *circuits of size cn,*

distinguish between the case where no assignments satisfy it, versus the case in which at least a $\geq 1 - 1/n^{10}$ fraction of the assignments satisfy it.

Lemma 14 follows from Theorem 13 almost directly: By Lemma 11, the class VSP_c (of functions recognizable by VSP circuits of size $\leq cn$) is efficiently closed under projections. Therefore, we can instantiate Theorem 13 on VSP_c. Since distinguishing the acceptance probability from being $= 1$ or $\leq 1/n^{10}$ is equivalent to distinguishing the *rejection* probability from being $= 0$ or $\geq 1 - 1/n^{10}$, we get Lemma 14 by *negating* the function which is AND of OR of $F_{n+O(\log n)}$ and using De Morgan's law on the AND. W.l.o.g. we replace the number of inputs $n + O(\log n)$ by n.

From Derandomizing VSP Circuits to Gap Block Disjointness

Let C be the circuit on n variables given as an input to the Derand-VSP problem described in Lemma 14. We use known results in complexity theory to convert this circuit into a simpler form that will be easier to work with when reducing to other problems. By Lemma 12, the circuit C can be interpreted as:

- OR of fan-in $n^{O(1)}$ of
- VSP circuits of size $\leq 4cn$,

where the $n^{O(1)}$ VSP circuits use the same set of n inputs.

Next, we use the following classical theorem of Valiant to convert each of these VSP circuits into an OR of CNF's on our n inputs. The ideas in the proof are due to Valiant [78], but the details were shown by Calabro [34] and Viola [81] (cf. Cygan et al. [46]).

▶ **Theorem 15** (Depth reduction [78]). *For all $\ell \geq 1$, we can convert any VSP of size $4cn$ on n variables into an equivalent formula which is OR of $2^{n/\ell}$ k-CNF's on the same n variables, where $k = 2^{2^{\mu c \ell}}$ for some absolute constant $\mu > 0$. The reduction runs in $2^{n/\ell} \cdot n^{O(1)}$ time for any constants c and l.*

▶ Remark. Let $\varepsilon > 0$ an arbitrary constant. Given a circuit on n variables with fan-in 2 gates, of size $O(n)$ and $O(\log n)$ depth, we can transform it into an equivalent formula which is *OR* of $2^{O(\frac{\log n}{\log \log n})}$ CNFs with clause size $\leq n^{\epsilon}$ [78]. However, we can't use this result for our purposes because it will be crucial for us that the clause size in the statement of Theorem 15 is upper bounded by a *constant*.

We will also need to apply the sparsification lemma [57, 58].

▶ **Lemma 16.** *For all $k \geq 3$ and $\varepsilon > 0$ we can convert a k-CNF formula on n variables into an equivalent OR of $2^{\varepsilon n}$ k-CNF formulas on the same variables where each CNF has $f(\varepsilon, k) \cdot n$ clauses, where $f(\varepsilon, k) = (k/\varepsilon)^{O(k)}$.*

Combining all these transformations allows us to focus on circuits of the following OR-AND-OR form. By the following claim, to solve the Derand-VSP problem it is enough to distinguish between the case in which no assignments satisfy a formula of the above OR-AND-OR form and the case in which at least $2^n - 2^n/n^{10}$ assignments do satisfy it.

▶ **Claim 17.** *Let C be an input circuit to the Derand-VSP problem (as described in Lemma 14). For all $\ell \geq 1$ and $\varepsilon > 0$, we can convert C into an* equivalent *formula C' on the same set of n inputs of the following form:*

- *OR of fan-in $n^{O(1)} \cdot 2^{n/\ell} \cdot 2^{\varepsilon n}$, of*
- *AND of fan-in $f(\varepsilon, k) \cdot n$ where $k = 2^{2^{\mu c \ell}}$, of*
- *OR of fan-in k of literals.*

Proof. Recall that an input circuit to the Derand-VSP problem has the form of an OR of fan-in $n^{O(1)}$ of series parallel circuits of size $\leq 4cn$. We want to decide if C is unsatisfiable or at least a $1 - 1/n^{10}$ fraction of the assignments satisfy it. First, we apply Theorem 15 on

every VSP circuit of size $\leq 4cn$. This produces a formula which is an OR of $2^{n/l}$ k-CNFs. Then, we apply the sparsification of Lemma 16 on every k-CNF to obtain a circuit as in the statement of the claim. ◀

This OR-AND-OR form motivates the definition of our Gap Block Disjointness problem (see Definition 1 in Section 1.1). Recall that our GBD Hypothesis (Hypothesis 2 in Section 1.1) states that GBD cannot be solved in truly subquadratic time with a deterministic algorithm. We are now ready to prove that refuting our hypothesis implies a circuit lower bound against linear size VSP circuits, thus establishing a "circuit lower bounds barrier" for refuting our hypothesis. The following claim implies Lemma 3 from Section 1.1.

▶ **Claim 18.** *For all $c \geq 1$ and $\alpha > 0$, there exists a constant $d \geq 1$ such that if there is a deterministic algorithm that solves the Gap Block Disjointness problem on two lists of size N of binary $N^\alpha \times d \log N$ matrices in $N^2/\log^{\omega(1)} N$ time, then E^{NP} does not have non-uniform VSP circuits of size cm (m is the number of input variables). The constant d can be upper bounded by*

$$d \leq 2^{2^{2^{O(c/\alpha)}}}.$$

Proof. By Theorem 13, to show that E^{NP} does not have non-uniform VSP circuits of size cm, it suffices to solve the Derand-VSP problem on a circuit C with $n = m + O(\log m)$ variables in time $2^n/n^{\omega(1)}$.

First, by Claim 17, we can transform the circuit C into an equivalent formula C' of form OR-AND-OR (as described in the statement). Then, we show a reduction from the Derand-VSP problem on the formula C' to the Block Disjoint Pairs problem with the required parameters, as follows. Let $N := 2^{n/2}$. We apply the transformation from Claim 17 to C, with parameters $\varepsilon := \frac{\alpha}{6}$, $l := \frac{6}{\alpha}$, and $d := 2f(\varepsilon, k) \leq (k/\varepsilon)^{O(k)}$, and get an equivalent formula C' of the following form:

- OR of fan-in $n^{O(1)} \cdot 2^{n/\ell} \cdot 2^{\varepsilon n} \leq 2^{\alpha n/2} = N^\alpha$, of
- AND of fan-in $f(\varepsilon, k) \cdot n = d \cdot \log N \leq (k/\alpha)^{O(k)} \cdot n$ where $k = 2^{2^{\mu c \ell}} \leq 2^{2^{O(c/\alpha)}}$, of
- OR of fan-in k of literals.

We think of the formula C' as a disjunction of CNF's with clause size k.

Let us now transform C' to an instance of the Block Disjoint Pairs problem. C' has n binary input variables $x_1, \ldots, x_n$. We split these variables into two parts: $x_1, \ldots, x_{n/2}$ and $x_{1+(n/2)}, \ldots, x_n$, and construct two sets A and B of matrices for the Block Disjoint Pairs problem.

Set of matrices A

Consider all the $N = 2^{n/2}$ partial assignments of the first half $x_1, \ldots, x_{n/2}$ of the variables. We will construct a matrix A_i, $i = 1, \ldots N$, for each partial assignment p_i of $x_1, \ldots, x_{n/2}$ as follows. For every k-CNF in C' we have a corresponding row in A_i, such that every clause of the k-CNF has a corresponding column. Thus, for $r = 1, \ldots, N^\alpha$, the r-th row of the matrix A_i corresponds to the r-th k-CNF in C', and every clause of the r-th k-CNF has a corresponding column in the r-th row such that the t-th clause corresponds to the t-th column in the r-th row of A_i. We set $A_i[r, t]$ to 0 if p_i satisfies the t-th clause of the r-th k-CNF, and to 1 otherwise. A clause is satisfied by a *partial* assignment iff it is satisfied independently of the assignment of the rest of the variables. We assume that the number of k-CNFs is N^α and the number of clauses in each k-CNF is $d \cdot \log N$. If this is not the case, then we can add dummy k-CNFs that are not satisfiable, or clauses that are satisfied by any partial assignment.

Set of matrices B

The second set of matrices B is constructed like the set A but with the second half of variables $x_{1+(n/2)}, \ldots, x_n$.

Our construction satisfies all the parameters of the Block Disjoint Pairs problem. In particular, $d \leq (k/\varepsilon)^{O(k)} \leq 2^{2^{2^{O(c/\alpha)}}}$.

Correctness of the reduction

To prove the correctness of our reduction, it suffices to show that the fraction of pairs of matrices that form a satisfying assignment (the first condition in Definition 1), is the same as the fraction of assignments that satisfy the circuit C'. We show that the i-th partial assignment of $x_1, \ldots, x_{n/2}$ and the j-th partial assignment of $x_{1+(n/2)}, \ldots, x_n$ satisfy C' iff the matrices A_i and B_j form a satisfying assignment too. If C' is satisfied, then at least one of the k-CNFs in C' is satisfied. Assume, then, without loss of generality, that the r-th k-CNF is satisfied. Our goal is to show that $\wedge_{h \in [d \log N]}(\neg A_i(r,h) \vee \neg B_j(r,h)) = True$. Or, equivalently, that $A_i(r,h) \cdot B_j(r,h) = 0$, for every $h \in [d \log N]$. In fact, this follows from the fact that the r-th k-CNF is satisfied and from the construction of A_i and B_j. If, on the other hand, $\vee_{k \in [N^\alpha]}(\wedge_{h \in [d \log N]}(\neg A_i(k,h) \vee \neg B_j(k,h))) = False$, then the i-th partial assignment of $x_1, \ldots, x_{n/2}$ and the j-th partial assignment of $x_{1+(n/2)}, \ldots, x_n$ do not satisfy C'. This follows from the construction of A_i and B_j and the fact that *no* k-CNF is satisfied in this case.

Therefore, $1 - 1/\log_2^{10} N = 1 - 2^{10}/n^{10} \leq 1 - 1/n^{10}$, concluding the proof. ◀

4 The Reduction to Approximate LCS

Our main technical contribution is a reduction from Gap Block Disjointness to $(1+\varepsilon)$ approximate LCS. Lemma 4 from Section 1.1 follows from the following claim, and the rest of this section is dedicated to its proof. For a high level intuition of the reduction see the introduction.

▶ **Claim 19** (Main). *If for some $\delta > 0$, there is a deterministic algorithm that can approximate the LCS of two given sequences of length n over an alphabet of size $n^{o(1)}$ to within a $(1+\varepsilon)$ factor, for all $\varepsilon > 0$, in $O(n^{2-\delta})$ time, then Hypothesis 2 is false.*

Weighted LCS

A natural generalization of LCS that will be useful in our proof is the *weighted longest common subsequence* (WLCS), where each symbol s has a positive integer weight $w(s)$. The weight of a subsequence is the total weight of the symbols it contains, and the WLCS score is the maximum total weight that we can obtain if we range over all common subsequences. As long as the weights are not too large, WLCS and LCS are computationally equivalent due the following lemma.

▶ **Lemma 20** (Lemma 2 in [2]). *Given two weighted sequences x and y, $WLCS(x,y) = LCS(x',y')$, where $|x'| = \sum_{i=1}^{|x|} w(x_i)$ and $|y'| = \sum_{i=1}^{|y|} w(y_i)$. The construction time of x', y' is $O(\max(|x'|,|y'|))$. $|z|$ denotes the length of the sequence z.*

Below, we will use the terms LCS and WLCS interchangeably, and if we do not specify the weight of a symbol s, then it is assumed that $w(s) = 1$.

The parameters of the reduction

We will show the following statement which implies what we need.

If we have a deterministic algorithm for the LCS problem that runs in $O(n^{2-\varepsilon})$ time and that gives $(1+(\delta/10^5))$ approximation, then we can solve the Block Disjoint Pairs problem in $O(N^{2-(\varepsilon/2)})$ time on binary matrices of size $N^\alpha \times d \log N$ for $\alpha := \varepsilon/10$ and $d := \alpha(1+\delta)/\delta$. Choosing $\delta = o(1)$ implies $d = \omega(1)$ and proves the theorem. W.l.o.g. we assume that $\delta \geq 1/\log n$ and $\varepsilon \leq 1/100$.

We will show the statement by providing a deterministic reduction from the Block Disjoint Pairs problem to the LCS problem. We will take the first set of N matrices $A = \{A_1, \ldots, A_N\}$ (each matrix is of size $N^\alpha \times d \log N$) and produce a sequence x of symbols. The sequence x is of length $|x| \leq O(N^{1+2\alpha}d/\alpha) =: n$. Similarly, we will take the second set of N matrices $B = \{B_1, \ldots, B_N\}$ and produce a sequence y of symbols, $|y| \leq O(N^{1+2\alpha}d/\alpha) = n$. The sequences x and y have the property that $LCS(x,y) \leq T$ if we are in Case 1 of Block Disjoint Pairs problem on sets of matrices A and B (see Definition 1) and $LCS(x,y) \geq (1+(\delta/10^5))T$ if we are in Case 2. T is some fixed value. We run the deterministic approximation algorithm for the LCS problem and decide in which case we are. Since the reduction is deterministic, this gives a deterministic algorithm for the Block Disjoint Pairs problem that runs in time

$$O(n^{2-\varepsilon}) \leq O\left(\left(N^{1+2\alpha}d/\alpha\right)^{2-\varepsilon}\right) \leq O\left(\left(N^{1+(\varepsilon/5)}/\delta\right)^{2-\varepsilon}\right),$$

where we use the fact that $\alpha = \varepsilon/10$ and $d = \alpha(1+\delta)/\delta$. Since $\delta \geq 1/\log n$ and $\varepsilon \leq 1/100$, we get that the runtime is upper bounded by $O(N^{(1+(\varepsilon/5))(2-\varepsilon)} \log^2 N) \leq O\left(N^{2-(\varepsilon/2)}\right)$ as required.

Construction of *inner gadgets*

To construct sequences x and y, we need *inner gadgets* $IG(A_i)$, $IG(B_j)$ for every set $A_i \in A$ and $B_j \in B$. We want that $IG(A_i)$ and $IG(B_j)$ satisfy the property that $LCS(IG(A_i), IG(B_j)) = T'$ if the pair A_i, B_j is not satisfying and $LCS(IG(A_i), IG(B_j)) = (1+\delta)T'$ if the pair A_i, B_j is satisfying. Below we will construct such inner gadgets with $|IG(A_i)|, |IG(B_j)| \leq O(N^{2\alpha}d/\alpha)$. After that we will construct the final sequences x and y with the required properties by putting together inner gadgets for all matrices in A and B.

We construct the inner gadgets $IG(A_i)$, $IG(B_j)$ by constructing *disjointness gadgets* $DG(A_i(k,\cdot))$, $DG(B_j(k,\cdot))$ for every row $k \in [K]$ of matrices A_i and B_j.

Properties of the disjointness gadgets

The disjointness gadgets will satisfy the following properties

- For every $k \in [K]$, $|DG(A_i(k,\cdot))|, |DG(B_j(k,\cdot))| \leq O(N^\alpha d/\alpha)$.
- If $\wedge_{h\in[D]}(\neg A_i(k,h) \vee \neg B_j(k,h)) = False$, then $LCS(DG(A_i(k,\cdot)), DG(B_j(k,\cdot))) = T'$. Otherwise, $LCS(DG(A_i(k,\cdot)), DG(B_j(k,\cdot))) = (1+\delta)T'$.
- For every $k \in [K]$, $DG(A_i(k,\cdot)), DG(B_j(k,\cdot)) \in \Sigma_k^*$. $\Sigma_k \cap \Sigma_{k'} = \emptyset$ for $k \neq k'$.

Given such disjointness gadgets, we construct inner gadgets $IG(A_i)$, $IG(B_j)$ as follows:

$$IG(A_i) := \bigcirc_{k=1}^{K} DG(A_i(k,\cdot)) = A_i(1,\cdot) \circ A_i(2,\cdot) \circ A_i(3,\cdot) \circ \ldots \circ A_i(k,\cdot),$$

$$IG(B_j) := \bigcirc_{k=1}^{K} DG(B_j(K+1-k,\cdot)) = B_j(k,\cdot) \circ B_j(k-1,\cdot) \circ B_j(k-2,\cdot) \circ \ldots \circ B_j(1,\cdot).$$

Since $\Sigma_k \cap \Sigma_{k'} = \emptyset$ for $k \neq k'$, the only way to get $LCS(IG(A_i), IG(B_j)) > 0$ is by matching symbols in $A_i(k,\cdot)$ and $B_j(k,\cdot)$. By the construction of $IG(A_i)$ and $IG(B_j)$, if we match

symbols between $A_i(k,\cdot)$ and $B_j(k,\cdot)$, then we can't match symbols between $A_i(k',\cdot)$ and $B_j(k',\cdot)$ if $k' \neq k$. This means that

$$LCS(IG(A_i), IG(B_j)) = \max_{k=1}^{K} LCS(DG(A_i(k,\cdot)), DG(B_j(k,\cdot))).$$

From the properties of disjointness gadgets, we get the required properties of the inner gadgets.

Construction of the disjointness gadgets

Now we will construct the disjointness gadgets $DG(A_i(k,\cdot)), DG(B_j(k,\cdot))$. $A_i(k,\cdot)$ is a binary vector of length $d \log N$. We split it into d/α binary vectors $v_t \in \{0,1\}^{\alpha \log N}$ each of length $|v_t| = \alpha \log N$: $A_i(k,\cdot) = v_1 \dots v_{d/\alpha}$. We define

$$DG(A_i(k,\cdot)) := c_k \circ \bigcirc_{t=1}^{d/\alpha} s_{k,t,v_t},$$

where we set $w(c_k) := (d/\alpha) - 1$. s_{k,t,v_t} are symbols indexed by rows k, indices of vectors t and vectors v_t. We have that

$$DG(A_i(k,\cdot)) \in \Sigma_k^* := \{c_k\} \cup \{s_{k,t,v} \mid v \in \{0,1\}^{\alpha \log N} \text{ and } t \in [d/\alpha]\}.$$

Similarly we split the binary vector $B_j(k,\cdot)$ of length $d \log N$ into d/α binary vectors $w_t \in \{0,1\}^{\alpha \log N}$ each of length $|w_t| = \alpha \log N$: $B_j(k,\cdot) = w_1 \dots w_{d/\alpha}$. We define

$$DG(B_j(k,\cdot)) := \left(\bigcirc_{t=1}^{d/\alpha} \bigcirc_{v \,:\, v \cdot w_t = 0} s_{k,t,v} \right) \circ c_k,$$

where we do the inner product $v \cdot w_t$ over the integers (not modulo 2). Notice that $|DG(A_i(k,\cdot))| \leq O(d/\alpha)$ and $|DG(B_j(k,\cdot))| \leq O(N^\alpha d/\alpha)$ as required.

We claim that if $\wedge_{h \in [D]}(\neg A_i(k,h) \vee \neg B_j(k,h)) = False$, then

$$LCS(DG(A_i(k,\cdot)), DG(B_j(k,\cdot))) = (d/\alpha) - 1$$

and $LCS(DG(A_i(k,\cdot)), DG(B_j(k,\cdot))) = d/\alpha$ otherwise. Since $d = \alpha(1+\delta)/\delta$, we have that $T' = (d/\alpha) - 1$ satisfies the second property of the disjointness gadgets. We now show the claim.

Clearly, $LCS(DG(A_i(k,\cdot)), DG(B_j(k,\cdot))) \geq (d/\alpha) - 1$ because we can match the symbols c_k. Also, we have the equality if we match the symbols c_k in the optimal alignment. Suppose that we don't match c_k. Then it's not hard to check that $LCS(DG(A_i(k,\cdot)), DG(B_j(k,\cdot))) = d/\alpha$ if $\wedge_{h \in [D]}(\neg A_i(k,h) \vee \neg B_j(k,h)) = True$ and $\leq (d/\alpha) - 1$ otherwise. Since we have to take maximum between the cases when we match the symbols c_k and when we don't match c_k, we get the required equalities.

The outer construction

In the remainder of the proof we construct the final sequences x and y with the promised properties. The sequence x is a concatenation of the inner gadgets $IG(A_i)$ with some additional symbols. The sequence y is a concatenation of the inner gadgets $IG(B_j)$ with some additional symbols. Each inner gadget $IG(B_j)$ appears twice in the sequence y.

We define integer values $v_0 < v_1 < v_2 < v_3$ as follows. We set $v_0 := T'$ (see the definition of the inner gadgets for T'), $v_1 := (1+\delta)T'$, $v_2 := 10v_1$, $v_3 := 100v_1$. For the simplicity of the notation, we will write A_i instead of $IG(A_i)$ and B_j instead of $IG(B_j)$. It will be clear

from the context whether we refer to A_i (B_j, resp.) or to $IG(A_i)$ ($IG(B_j)$, resp.). We define the sequence x:

$$x := \left(\bigcirc_{i=1}^{3n} 2\right) \circ \left(\bigcirc_{i=1}^{n} (0\ A_i\ 1)\right) \circ \left(\bigcirc_{i=1}^{3n} 2\right).$$

We define the sequence y:

$$y := \left(\bigcirc_{j=1}^{n} (2\ 0\ B_j\ 1)\right) \circ \left(\bigcirc_{j=1}^{n} (2\ 0\ B_j\ 1)\right) \circ 2.$$

We set the weight of symbols 0, 1 and 2 as follows: $w(2) := v_2$ and $w(0) = w(1) := v_3$.

We have two goals. First, we want to show that if there are many satisfying assignments (each assignment is a pair of A_i and B_j), then the LCS score between x and y is large: $LCS(x, y) \geq (1 + (\delta/10^5))T$. T is some fixed value that we will define later. Second, if there are no satisfying assignments, then the LCS score is small: $LCS(x, y) \leq T$. We achieve these two goals via the next two lemmas.

▶ **Lemma 21.** *If there are many satisfying assignment (see Definition 1), then*

$$LCS(x, y) \geq (n + 2)v_2 + 2nv_3 + 0.99nv_1 + 0.01nv_0 =: T''.$$

Proof. We will exhibit n different alignments between x and y and we will show that at least one of them achieves the LCS score T''. This gives the lower bound on $LCS(x, y)$.

For $k = 1, \ldots, n$ we write

$$y = \left(\bigcirc_{j=1}^{k-1} (2\ 0\ B_j\ 1)\right) \circ 2 \circ s_k \circ \left(\bigcirc_{j=k}^{n} (2\ 0\ B_j\ 1)\right) \circ 2,$$

where

$$s_k := (0\ B_k\ 1) \circ \left(\bigcirc_{j=k+1}^{n} (2\ 0\ B_j\ 1)\right) \circ \left(\bigcirc_{j=1}^{k-1} (2\ 0\ B_j\ 1)\right).$$

Clearly,

$$\begin{aligned} LCS(x, y) \geq & LCS\left(\bigcirc_{i=1}^{3n} 2, \left(\bigcirc_{j=1}^{k-1} (2\ 0\ B_j\ 1)\right) \circ 2\right) \\ & + LCS\left(\bigcirc_{i=1}^{n} (0\ A_i\ 1), s_k\right) \\ & + LCS\left(\bigcirc_{i=1}^{3n} 2, \left(\bigcirc_{j=k}^{n} (2\ 0\ B_j\ 1)\right) \circ 2\right). \end{aligned}$$

The total contribution of the first and the third term on the r.h.s. is $(n + 2)v_2$ because only symbols 2 can contribute to the LCS score and there are $n + 2$ symbols 2. For the middle term we align inner gadgets in pairs and match all symbols 0 and 1. We get the lower bound

$$LCS(x, y) \geq (n + 2)v_2 + 2nv_3 + \sum_{i=1}^{n} LCS(A_i, B_{j_k(i)}),$$

$$where j_k(i) := \begin{cases} i + k - 1 & \text{if } i \leq n + 1 - k, \\ i + k - 1 - n & \text{otherwise.} \end{cases}$$

By averaging the r.h.s. over all $k = 1, \ldots, n$, we get

$$\begin{aligned} LCS(x, y) &\geq \frac{1}{n} \sum_{k=1}^{n} \left((n + 2)v_2 + 2nv_3 + \sum_{i=1}^{n} LCS(A_i, B_{j_k(i)}) \right) \\ &= (n + 2)v_2 + 2nv_3 + \frac{1}{n} \sum_{i,k=1}^{n} LCS(A_i, B_{j_k(i)}) \\ &= (n + 2)v_2 + 2nv_3 + \frac{1}{n} \sum_{i,j=1}^{n} LCS(A_i, B_j) \\ &\geq (n + 2)v_2 + 2nv_3 + 0.99nv_1 + 0.01nv_0, \end{aligned}$$

where in the last inequality we use the fact that there are many satisfying assignments. This finishes the proof of the lemma. ◀

▶ **Lemma 22.** *If there are no satisfying assignments, then*

$$LCS(x,y) \leq (n+2)v_2 + 2nv_3 + nv_0 =: T.$$

Proof. We start with the intuition behind the analysis.

Intuition

We saw in the proof of Lemma 21 that there is an alignment that achieves a large LCS score. In the alignment we match the n inner gadgets from the first sequence x with an n consecutive inner gadgets from the second sequence y in pairs. We want to claim that in an optimal alignment, we will do the same: map the n inner gadgets from the first sequence with an n consecutive inner gadgets from the second sequence in pairs. Intuitively, this is because of the following three reasons:

- We don't want to choose less than n inner gadgets from the second sequence because otherwise we can't match all symbols 0 and 1 from the first sequence with their counterparts (symbols 0 and 1 have the largest weight - we loose a lot by not matching them).
- We don't want to choose more than n inner gadgets from the second sequence because otherwise we have fewer symbols 2 from the second sequence to be matched with their counterparts. Symbols 2 have smaller weight than symbols 0 and 1 but still we loose a lot by not matching them.
- Finally, if we choose n inner gadgets from the second sequence we want to match them in pairs. If we don't do that, we can't match all symbols 0 and 1 which is again expensive.

We proceed to formalize the intuition.

Sequence x starts with $3n$ copies of symbol 2. Suppose that some of those symbols are matched. W.l.o.g. the matched symbols from a suffix of $\bigcirc_{i=1}^{3n} 2$. W.l.o.g. the last symbol of $\bigcirc_{i=1}^{3n} 2$ is matched. If this is not the case we can match it with the first symbol of y and this can only increase LCS. Consider the symbol 2 from y that is matched to the last symbol 2 from $\bigcirc_{i=1}^{3n} 2$. Consider the symbol to the right of the symbol 2 in y. It is 0. Let s be its position in y. W.l.o.g. this symbol 0 is matched to the first symbol 0 from x. If this is not so, we can make this match and this can't decrease the LCS score. Analogously we can argue that the last symbol 1 from x is matched to a symbol 1 in y. Let t be the location of the symbol 1 in y. Let x' be the substring of x that is to the right of the first symbol 0 in x and to the left of the last symbol 1 in x. Let y' be the substring of y that is to the right of the symbol 0 at location s in y and to the left of the symbol 1 at location t in y. We write $x = x_1 x' x_2$ and $y = y_1 y' y_2$. We can upper bound LCS if we range over all such partitions of x and y:

$$LCS(x,y) \leq \max_{\substack{x=x_1x'x_2 \\ y=y_1y'y_2}} LCS(x_1,y_1) + LCS(x',y') + LCS(x_2,y_2). \tag{1}$$

Let $m \geq 1$ denote the number of inner gadgets in y'.

▶ **Claim 23.**

$$LCS(x_1,y_1) + LCS(x_2,y_2) \leq 2v_3 + (2n+1-(m-1))v_2.$$

Proof. The total number of symbols 2 in y_1 and y_2 is $2n+1-(m-1)$. The total contribution from all symbols 2 is upper bounded by $(2n+1-(m-1))v_2$. We can also match symbol 0 in x_1 and symbol 1 in x_2. This upper bounds the total contribution from symbols 0 and 1 by $2v_3$. There are no other symbols that we can match. The claim follows. ◀

It remains to give an upper bound on $LCS(x', y')$. Consider any symbol 0 in x'. Its *neighbour* is the symbol 1 to the left of it. Similarly, for any symbol 1 in x' the neighbour is the symbol 0 to the right of it. For any symbol 0 in y' the neighbour is the first symbol 1 to the left of it. For any symbol 1 in y' the neighbour is the first symbol 0 to the right of it. For any two symbols 0 that are matched between x' and y', their neighbours (symbols 1) form a match too. If this does not true, we match the symbols 1 and this can only increase the LCS score. Similarly, for any two symbols 1 that are matched, their neighbours form a match too. Let $M \geq 0$ denote the number of pairs of matched symbols 0 and 1. This allows us to upper bound the total contribution from symbols 0 and 1 to $LCS(x', y')$ by $S := 2Mv_2$. The M pairs of matched symbols split the sequence x into $M+1$ maximal substrings $r_1, \ldots, r_{M+1}$. In each one of the $M+1$ substrings s_i does not contain a symbol 0 or 1 that is matched. Similarly, we split y' into $M+1$ maximal substrings $p_1, \ldots, p_{M+1}$ so that each p_i does not contain a symbol 0 or 1 that is matched. Symbols in r_i can only be matched to symbols in p_i. The only symbols that can be matched from r_i with symbols from p_i come from the inner gadgets by the definition of r_i and p_i. Let $d_i \geq 1$ denote the number of the inner gadgets in r_i and $l_i \geq 1$ denote the number of the inner gadgets in p_i. Clearly, $\sum_{i=1}^{M+1} d_i = n$ and $\sum_{i=1}^{M+1} l_i = m$. We claim that $LCS(r_i, p_i) \leq (d_i + l_i - 1)v_0$. Since the pairs of matched symbols can't cross, we can easily check that the total number of pairs of inner gadgets that can have a match is upper bounded by $d_i + l_i - 1$. Because there are no satisfying assignments, the upper bound $LCS(r_i, p_i) \leq (d_i + l_i - 1)v_0$ follows. From all this we have

$$LCS(x', y') \leq S + \sum_{i=1}^{M+1} LCS(r_i, p_i) \leq 2Mv_2 + \sum_{i=1}^{M+1} (d_i + l_i - 1)v_0.$$

We combine this with the equalities $\sum_{i=1}^{M+1} d_i = n$ and $\sum_{i=1}^{M+1} l_i = m$ and get the following Claim.

▶ **Claim 24.**

$$LCS(x', y') \leq 2Mv_3 + (n+m)v_0 - (M+1)v_0.$$

We combine (1) with Claims 23 and 24 and get the following upper bound:

$$\begin{aligned} LCS(x, y) \leq & 2v_2 + n(2v_2 + v_0) \\ & + (M+1)(2v_3 - v_0) - m(v_2 - v_0). \end{aligned}$$

From $\sum_{i=1}^{M+1} d_i = n$ and $\sum_{i=1}^{M+1} l_i = m$ we get that $M \leq \min(n, m) - 1$. As we increase M, the r.h.s. of the upper bound only increases. We choose $M = \min(n, m) - 1$. Consider two cases.

- $m \geq n$. We have $M = n - 1$ and $LCS(x, y) \leq v_2(2n+2) + 2nv_3 - m(v_2 - v_0) \leq T$.
- $m \leq n$. We have $M = m - 1$ and $LCS(x, y) \leq v_2(2n+2) + nv_0 + m(2v_3 - v_2) \leq T$. ◀

From the above Lemmas 21 and 22 we have that $LCS(x, y) \geq T''$ if there are many satisfying assignment and $LCS(x, y) \leq T$ if there are no satisfying assignments. From the definition of values v_0, v_1, v_2, v_3 (in particular, $v_1 = (1+\delta)v_0$), we can easily conclude that $T'' \geq (1 + (\delta/10^5))T$ which gives the properties of x and y that we need.

Harder variants of the Block Disjoint Pairs problem

In the paragraph "Construction of the disjointness gadgets", we do the following construction. Given two vectors $z^k := A_i(k,\cdot), w^k := B_j(k,\cdot) \in \{0,1\}^{d \log N}$, we construct sequences $DG(z^k)$ and $DG(w^k)$ such that LCS between them is $LCS(z^k, w^k) = d/\alpha$ if the vectors are orthogonal and $LCS(z^k, w^k) = (d/\alpha) - 1$ otherwise. We split the vector z^k into d/α shorter vectors $z^k_1, \ldots, z^k_{d/\alpha} \in \{0,1\}^{\alpha \log N}$. Similarly, we split the vector w^k into d/α shorter vectors $w^k_1, \ldots, z^k_{d/\alpha} \in \{0,1\}^{\alpha \log N}$. We construct $DG(z^k)$ by replacing each shorter vector z^k_t by a symbol corresponding to it (indexed by the $\alpha \log N$ binary values) and its position. We construct $DG(w^k)$ by replacing each shorter vector w^k_t by a sequence of symbols corresponding to all vectors that are orthogonal to w^k_t. This implies that we have a large LCS score if there are many orthogonal pairs z^k_t, w^k_t of short vectors. Instead of replacing w^k_t by a sequence of symbols corresponding to all orthogonal vectors, we can take an arbitrary function $f^k_t : \{0,1\}^{2\alpha \log N} \to \{0,1\}$ and replace w^k_t by a sequence of symbols corresponding to all vectors $u \in \{0,1\}^{\alpha \log N}$ such that $f^k_t(u, w^k_t) = 1$. We recover the orthogonality constraint by choosing functions f^k_t that evaluates to 1 iff the two vectors are orthogonal. For arbitrary functions $f^k_1, \ldots, f^k_{d/\alpha} : \{0,1\}^{2\alpha \log N} \to \{0,1\}$, we get that $LCS(z^k, w^k) = d/\alpha$ if $f^k_1(z^k_1, w^k_1) = \ldots = f^k_{d/\alpha}(z^k_{d/\alpha}, w^k_{d/\alpha}) = 1$ and $LCS(z^k, w^k) = (d/\alpha) - 1$ otherwise. Clearly, the new version of Block Disjoint Pairs problem is harder to solve than the one restricted to the orthogonality constraints.

To further increase the hardness of the Block Disjoint Pairs problem we can define functions $g^k : \{0,1\}^{d/\alpha} \to \{0,1\}$ and require that $LCS(z^k, w^k) = q$ if

$$g^k(f^k_1(z^k_1, w^k_1), \ldots, f^k_{d/\alpha}(z^k_{d/\alpha}, w^k_{d/\alpha})) = 1$$

and $LCS(z^k, w^k) = q' < q$ otherwise (for some constants q and q'). Notice that previously all functions g^k are AND functions. This modification requires that the gap $(q/q') - 1$ is at least a constant (we have a constant gap for the AND function) and that the functions g^k can be efficiently simulated with LCS.

4.1 Hardness for Approximate Binary LCS and Edit Distance

The results in this section follow from simple observations over [5] that are easy to make with our framework in mind.

We refer the reader to [5] for the definition and background on Branching Programs.

▶ **Theorem 25** (Theorem 2 in [5]). *There is a reduction from SAT on nondeterministic branching programs on m variables, length T, and width W, to an instance of Edit-Distance or LCS on two binary sequences x and y of length $n = 2^{m/2} \cdot T^{O(\log W)}$, and the reduction runs in $O(n)$ time.*

We need additional properties of the reduction from Theorem 25.

▶ **Claim 26.** *Let P be the Branching Program that we want to reduce.*

If we reduce Branching Program P to LCS problem then we have the following two properties:

- *If P is not satisfiable, then $LCS(x,y) \leq C$ for some integer constant $C = C(m,T,W) \leq n$.*
- *If at least half of the assignments satisfy the Branching Program P, then $LCS(x,y) \geq C + (2^{m/2}/2)$.*

If we reduce Branching Program P to Edit-Distance problem then we have the following two properties:

- *If P is not satisfiable, then $Edit(x,y) \geq C$ for some integer constant $C = C(m,T,W) \leq n$.*

- *If at least half of the assignments satisfy the Branching Program P, then $Edit(x, y) \leq C - (2^{m/2}/2)$.*

Proof. The proof follows from the proof of Claim 9 in [5].

Consider the case when P is not satisfiable. The proof does not change - we show that LCS is upper bounded and Edit-Distance is lower bounded by some fixed quantity C.

Consider the case when P is satisfied by at least half of the assignments. In the proof of Claim 9 the authors choose an integer Δ such that the corresponding alignment pairs up two gadgets that form a satisfying assignment to the Branching Program P. When there are many satisfying assignments (at least half), we can show that there is an integer such in the corresponding alignment at least half of the assignments are satisfying. By the properties of the gadgets constructed in [5], we get the required lower bound on LCS and the required upper bound on Edit-Distance. ◀

Theorem 25 and Claim 26 combined give the following theorem.

▶ **Theorem 27.** *Suppose we have a $(1 + \delta)$ approximation algorithm for Edit-Distance or LCS with $\delta = o(1/T^{O(\log W)})$ that runs in $f(n) = f(2^{m/2} \cdot T^{O(\log W)})$ deterministic time for some function f. Then in time $f(2^{m/2} \cdot T^{O(\log W)})$ we can decide if a Branching program on m variables, length T and width W is not satisfiable or at last half of the assignments are satisfying.*

From the discussion in [5] on the connection between BPs and NC circuits, a lower bound for NC^1 follows. Namely, $1 + 1/\text{poly} \log n$ approximation algorithm implies that there exists $f \in \mathsf{E}^{\mathsf{NP}}$ such that $f \notin NC^1$.

Acknowledgments. We thank Piotr Indyk, Michael P. Kim, Dana Moshkovitz, Virginia Vassilevska Williams, and Ryan Williams for helpful discussions on this work.

References

1 Amir Abboud, Arturs Backurs, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Or Zamir. Subtree isomorphism revisited. In *Proc. of 27th SODA*, pages 1256–1271, 2016.

2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight Hardness Results for LCS and other Sequence Similarity Measures. In *Proc. of 56th FOCS*, pages 59–78, 2015.

3 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 98–117. IEEE, 2015.

4 Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proc. of 26th SODA*, pages 1681–1697, 2015.

5 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. In *STOC'16*, 2016.

6 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. of 55th FOCS*, pages 434–443, 2014.

7 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proc. of 27th SODA*, pages 377–391, 2016.

8 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster sequence alignment. In *Proc. of 41st ICALP*, pages 39–51, 2014.

9 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proc. of 47th STOC*, pages 41–50, 2015.

10 Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proc. of 26th SODA*, pages 218–230, 2015.

11 Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter. *arXiv preprint arXiv:1506.01799*, 2015.

12 Josh Alman, Timothy M Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *to appear at FOCS*, 2016.

13 Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

14 A. Amir, T. M. Chan, M. Lewenstein, and N. Lewenstein. On hardness of jumbled indexing. In *Proc. ICALP*, volume 8572, pages 114–125, 2014.

15 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *FOCS*, pages 377–386, 2010.

16 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. Bpp has subexponential time simulations unless exptime has publishable proofs. In *Structure in Complexity Theory Conference, 1991., Proceedings of the Sixth Annual*, pages 213–219. IEEE, 1991.

17 Rolf Backofen, Dekel Tsur, Shay Zakov, and Michal Ziv-Ukelson. Sparse rna folding: Time and space efficient algorithms. *Journal of Discrete Algorithms*, 9(1):12–31, 2011.

18 Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. *arXiv preprint arXiv:1602.05837*, 2016.

19 Arturs Backurs and Piotr Indyk. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false). In *Proc. of 47th STOC*, pages 51–58, 2015.

20 Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? *arXiv preprint arXiv:1511.07070*, 2015.

21 Ziv Bar-Yossef, TS Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 550–559. IEEE, 2004.

22 Tuğkan Batu, Funda Ergun, and Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 792–801. Society for Industrial and Applied Mathematics, 2006.

23 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Short pcps verifiable in polylogarithmic time. In *Computational Complexity, 2005. Proceedings. Twentieth Annual IEEE Conference on*, pages 120–134. IEEE, 2005.

24 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust pcps of proximity, shorter pcps and applications to coding. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 1–10, 2004. doi:10.1145/1007352.1007361.

25 Eli Ben-Sasson and Madhu Sudan. Short pcps with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008.

26 Eli Ben-Sasson and Emanuele Viola. Short pcps with projection queries. In *ICALP, Part I*, pages 163–173, 2014.

27 Lasse Bergroth, Harri Hakonen, and Timo Raita. New approximation algorithms for longest common subsequences. In *String Processing and Information Retrieval: A South American Symposium, 1998. Proceedings*, pages 32–40. IEEE, 1998.

28 Lasse Bergroth, Harri Hakonen, and Timo Raita. A survey of longest common subsequence algorithms. In *String Processing and Information Retrieval, 2000. SPIRE 2000. Proceedings. Seventh International Symposium on*, pages 39–48. IEEE, 2000.

29 Mark Braverman, Young Kun-Ko, and Omri Weinstein. Approximating the best nash equilibrium in $n^{o(\log n)}$-time breaks the exponential time hypothesis. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 970–982, 2015. doi:10.1137/1.9781611973730.66.

30 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless seth fails. In *Proc. of 55th FOCS*, pages 661–670, 2014.

31 Karl Bringmann and Marvin Künnemann. Improved approximation for fréchet distance on c-packed curves matching conditional lower bounds. *CoRR*, abs/1408.1340, 2014. URL: http://arxiv.org/abs/1408.1340.

32 Karl Bringmann and Marvin Kunnemann. Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping. In *Proc. of 56th FOCS*, pages 79–97, 2015.

33 Karl Bringmann and Wolfgang Mulzer. Approximability of the Discrete Fréchet Distance. In *Proc. of 31st SoCG*, pages 739–753, 2015.

34 Chris Calabro. A lower bound on the size of series-parallel graphs dense in long paths. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 15, 2008.

35 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *Proc. of 4th IWPEC*, pages 75–85, 2009.

36 Marco L Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 261–270. ACM, 2016.

37 Marco L Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Tighter connections between derandomization and circuit lower bounds. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 40. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

38 Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, 2015.

39 Timothy M Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA, 2016.

40 Yi-Jun Chang. Hardness of rna folding problem with four symbols. *arXiv preprint arXiv:1511.04731*, 2015.

41 Krishnendu Chatterjee, Wolfgang Dvořák, Monika Henzinger, and Veronika Loitzenbauer. Model and objective separation with conditional lower bounds: Disjunction is harder than conjunction. *arXiv preprint arXiv:1602.02670*, 2016.

42 Shiri Chechik, Daniel H Larkin, Liam Roditty, Grant Schoenebeck, Robert E Tarjan, and Virginia Vassilevska Williams. Better approximation algorithms for the graph diameter. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1041–1052. SIAM, 2014.

43 F Chin and Chung Keung Poon. Performance analysis of some simple heuristics for computing longest common subsequences. *Algorithmica*, 12(4-5):293–311, 1994.

44 Thomas H.. Cormen, Charles Eric Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*, volume 6. MIT press Cambridge, 2001.

45 Maxime Crochemore, Costas S Iliopoulos, Yoan J Pinzon, and James F Reid. A fast and practical bit-vector algorithm for the longest common subsequence problem. *Information Processing Letters*, 80(6):279–285, 2001.

46 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as cnf-sat. In *Computational Complexity (CCC), 2012 IEEE 27th Annual Conference on*, pages 74–84. IEEE, 2012.

47 Søren Dahlgaard. On the hardness of partially dynamic graph problems and connections to diameter. *arXiv preprint arXiv:1602.06705*, 2016.

48 J Boutet de Monvel. Extensive simulations for longest common subsequences. *The European Physical Journal B-Condensed Matter and Complex Systems*, 7(2):293–308, 1999.

49 Robert C Edgar and Serafim Batzoglou. Multiple sequence alignment. *Current opinion in structural biology*, 16(3):368–373, 2006.

50 Lance Fortnow and Adam R Klivans. Efficient learning algorithms yield circuit lower bounds. *Journal of Computer and System Sciences*, 75(1):27–36, 2009.

51 Yelena Frid and Dan Gusfield. A simple, practical and complete o-time algorithm for rna folding using the four-russians speedup. *Algorithms for Molecular Biology*, 5(1):1, 2010.

52 A. Gajentaan and M. H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 45(4):140–152, 2012.

53 Ofer Grossman and Dana Moshkovitz. Amplification and derandomization without slowdown. *arXiv preprint arXiv:1509.08123*, 2015.

54 Dan Gusfield. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge university press, 1997.

55 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 21–30. ACM, 2015.

56 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

57 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

58 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63:512–530, 2001.

59 Russell Impagliazzo and Avi Wigderson. P= bpp if e requires exponential circuits: Derandomizing the xor lemma. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 220–229. ACM, 1997.

60 Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *Automata, Languages, and Programming*, pages 749–760. Springer, 2015.

61 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

62 Richard M Karp and Richard Lipton. Turing machines that take advice. *Enseign. Math*, 28(2):191–209, 1982.

63 Kazutaka Katoh and Daron M Standley. Mafft multiple sequence alignment software version 7: improvements in performance and usability. *Molecular biology and evolution*, 30(4):772–780, 2013.

64 Adam Klivans, Pravesh Kothari, and Igor C Oliveira. Constructing hard functions using learning algorithms. In *Computational Complexity (CCC), 2013 IEEE Conference on*, pages 86–97. IEEE, 2013.

65 Gad M Landau, Eugene W Myers, and Jeanette P Schmidt. Incremental string comparison. *SIAM Journal on Computing*, 27(2):557–582, 1998.

66 William J Masek and Michael S Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System sciences*, 20(1):18–31, 1980.

67 Thilo Mie. Short pcpps verifiable in polylogarithmic time with o (1) queries. *Annals of Mathematics and Artificial Intelligence*, 56(3-4):313–338, 2009.

68 Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

69 Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. *Journal of the ACM (JACM)*, 54(5):23, 2007.

70 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.

71 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. of 45th STOC*, pages 515–524, 2013.

72 Balaram Saha. The dyck language edit distance problem in near-linear time. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 611–620. IEEE, 2014.

73 Barna Saha. Language edit distance and maximum likelihood parsing of stochastic grammars: Faster algorithms and connection to fundamental graph problems. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 118–135. IEEE, 2015.

74 Rajesh Santhanam and Ross Williams. On medium-uniformity and circuit lower bounds. In *Computational Complexity (CCC), 2013 IEEE Conference on*, pages 15–23. IEEE, 2013.

75 Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

76 Yinglei Song. Time and space efficient algorithms for rna folding with the four-russians technique. *arXiv preprint arXiv:1503.05670*, 2015.

77 Julie D Thompson, Desmond G Higgins, and Toby J Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, 22(22):4673–4680, 1994.

78 Leslie G Valiant. *Graph-theoretic arguments in low-level complexity*. Springer, 1977.

79 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proc. of 51st FOCS*, pages 645–654, 2010.

80 Balaji Venkatachalam, Dan Gusfield, and Yelena Frid. Faster algorithms for rna-folding using the four-russians method. *Algorithms for Molecular Biology*, 9(1):1, 2014.

81 Emanuele Viola. *On the power of small-depth computation*. Now Publishers Inc, 2009.

82 Ryan Williams. A new algorithm for optimal constraint satisfaction and its implications. In *Automata, Languages and Programming*, pages 1227–1237. Springer, 2004.

83 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013.

84 Ryan Williams. Natural proofs versus derandomization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 21–30. ACM, 2013.

85 Ryan Williams. Algorithms for Circuits and Circuits for Algorithms: Connecting the Tractable and Intractable. In *Proceedings of the International Congress of Mathematicians*, 2014. URL: `http://web.stanford.edu/~{}rrwill/ICM-survey.pdf`.

86 Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014.

87 Ryan Williams. Strong ETH Breaks With Merlin and Arthur: Short Non-Interactive Proofs of Batch Evaluation. In *CCC'16*, 2016.