Graphs Cannot Be Indexed in Polynomial Time for Sub-quadratic Time String Matching, Unless SETH Fails

Equi, Massimo

Springer
2021

Equi , M , Mäkinen , V & Tomescu , A 2021 , Graphs Cannot Be Indexed in Polynomial Time þÿfor Sub-quadratic Time String Matching, Unless SETH Fails . in T Bure
Gamper , G Guerrini , T Jurdzinski , C Pahl , F Sikora & P W Wong (eds) , SOFSEM 2021: Theory and Practice of Computer Science. SOFSEM 2021 . Lecture Notes in Computer Science , vol. 12607 , Springer , Cham , pp. 608-622 , 47th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2021 , Bolzano-Bozen , Italy , 25/01/2021 . https://doi.org/10.1007/978-3-030-67731-2_44

http://hdl.handle.net/10138/345768
https://doi.org/10.1007/978-3-030-67731-2_44

acceptedVersion

Graphs cannot be indexed in polynomial time for sub-quadratic time string matching, unless SETH fails

Massimo Equi, Veli Mäkinen, and Alexandru I. Tomescu

Department of Computer Science, University of Helsinki
{massimo.equi,veli.makinen,alexandru.tomescu}@helsinki.fi

March 5, 2020

Abstract

We consider the following string matching problem on a node-labeled graph $G = (V, E)$: given a pattern string P, decide whether there exists a path in G whose concatenation of node labels equals P. This is a basic primitive in various problems in bioinformatics, graph databases, or networks. The hardness results of Backurs and Indyk (FOCS 2016) imply that this problem cannot be solved in better than $O(|E||P|)$ time, under the Orthogonal Vectors Hypothesis (OVH), and this holds even under various restrictions on the graph (Equi et al., ICALP 2019).

In this paper we consider its *offline* version, namely the one in which we are allowed to index the graph in order to support time-efficient string matching queries. In fact, this version is the one relevant in practical application such as the ones mentioned above. While the online version has been believed to be hard even before the above-mentioned hardness results, it was tantalizing in the string matching community to believe that the offline version can allow for sub-quadratic time queries, e.g. at the cost of a high-degree polynomial-time indexing.

We disprove this belief, showing that, under OVH, no polynomial-time indexing scheme of the graph can support querying P in time $O(|E|^{\delta}|P|^{\beta})$, with either $\delta < 1$ or $\beta < 1$. We prove this *tight* bound employing a known self-reducibility technique, e.g. from the field of dynamic algorithms, which translates conditional lower bounds for an online problem to its offline version.

As a side-contribution, we formalize this technique with the notion of *linear independent-components reduction*, allowing for a simple proof of our result. As another illustration that hardness of indexing follows as a corollary of a linear independent-components reduction, we also translate the quadratic conditional lower bound of Backurs and Indyk (STOC 2015) for the problem of matching a query string inside a text, under edit distance. We obtain an analogous *tight* quadratic lower bound for its offline version, improving the recent result of Cohen-Addad, Feuilloley and Starikovskaya (SODA 2019), but with a slightly different boundary condition.

1 Introduction

1.1 Background

The *String Matching in Labeled Graphs (SMLG)* problem is defined as follows.

Problem 1 (SMLG)**.**

INPUT: A directed graph $G = (V, E, \ell)$, where each node $v \in V$ is labeled by a character $\ell(v)$, and a pattern string P.

OUTPUT: *True* if and only if there is path $(v_1, v_2, \ldots, v_{|P|})$ in G such that $P[i] = \ell(v_i)$ holds for all $1 \leq i \leq |P|$.

This is a natural generalization of the problem of matching a string inside a text, and it is a primitive in various problems in computational biology, graph databases, and graph mining. In genome research, the very first step of many standard analysis pipelines of high-throughput sequencing data is nowadays to align sequenced fragments of DNA on a labeled graph (a so-called *pan-genome*) that encodes all genomes of a population [41, 16, 32, 26]. In graph databases, query languages provide the user with the ability to select paths based on the labels of their nodes or edges [9, 24, 40, 37]. In graph mining, this is a basic ingredient related to computing graph kernels [30] or node similarity [17].

The SMLG problem can be solved in time $O(|V| + |E||P|)$ [7] in the comparison model. On acyclic graphs, bitparallelism can be used for improving the time to $O(|V| + |E|\lceil|P|/w\rceil)$ [38] in the RAM model with word size $w = \Theta(\log |E|)$. It remained an open question whether a truly sub-quadratic time algorithm for it exists. However, the recent conditional lower bounds by Backurs and Indyk [11] for regular expression matching imply that the SMLG problem cannot be solved in sub-quadratic time, unless the so-called *Orthogonal Vectors Hypothesis (OVH)* is false. This result was strengthened by Equi et al. [20] by showing that the problem remains quadratic under OVH also on directed acyclic graphs (DAGs), that are even *deterministic*, in the sense that for every node, the labels of its out-neighbors are all distinct.

As mentioned above, in real-world applications one usually considers the *offline* version of the SMLG problem. Namely, we are allowed to index the labeled graph so that we can query for pattern strings in possibly sub-quadratic time. In the case when the graph is just a labeled (directed) path, then the problem asks about indexing a text string, which is a fundamental problem in string matching. There exists a variety of indexes constructable in *linear time* supporting *linear-time* queries [18]. The same holds also when the graph is a tree [23]. A trivial indexing scheme for arbitrary graphs is to enumerate all the possibly exponentially many paths of the graph and index those as strings. So a natural question is whether we can at least index the graph in polynomial time to support sub-quadratic time queries. Note that the conditional lower bound for the online problem naturally refutes the possibility of an index constructable in sub-quadratic time to support sub-quadratic time queries. Even before the OVH-based reductions, another weak lower bound was known to hold conditioned on the hardness of indexing for set intersection queries [12] (see also Table 1). We discuss this connection to the *Set Intersection Conjecture (SIC)* [36, 28] in Appendix A.

The connections to SIC and to OVH constrain the possible construction and query time tradeoffs for SMLG, but they are yet not strong enough to prove the impossibility of building an index in polynomial time such that queries could be sub-quadratic, or even take time say $O(|E|^{1/2}|P|^2)$. This would be a significant result. In fact, given the wide applicability of this problem, there have been many attempts to obtain such indexing schemes. Sirén, Välimäki, and Mäkinen [43] proposed an extension of the *Burrows-Wheeler transform* [14] for prefix-sorted graphs. Standard indexing techniques [29, 22, 35] can be applied on such generalized Burrows-Wheeler transform to support linear time pattern search. The bottleneck of the approach is the prefix-sorting step, which requires finding shortest prefixes for all paths such that they distinguish the nodes from each other. The size of the transform is still exponential in the worst case. However, unlike the trivial indexing scheme, it is linear in the best case, and also linear in the average case under a realistic model for genomics applications [43]. There have been some advances in making the approach more practical [42, 32, 26], but the exponential bottleneck has remained. Since in real-world scenarios approximate search is required on the graph, there have also been advances in expanding sparse dynamic programming and chaining algorithms [33], as well as the seed-and-extend strategy [19, 39] to this setting.

The concept of prefix-sorted graphs was later formalized into a more general concept of *Wheeler graphs* [25]: Conceptually these are a class of graphs that admit a generalization of the Burrows-Wheeler transform, and thus an index of size linear in the size of the graph, supporting string search in linear time in the size of the query pattern. Gibney and Thankachan showed that Wheeler graph recognition problem is NP-complete [27]. Alanko et al. [5] give polynomial time solutions on some special cases and improve the prefix-sorting algorithm to work in near-optimal time in the size of the output. They also give an example where such output can be of exponential size even for *acyclic deterministic finite automata* (acyclic DFA). One could conjecture that conversion of a graph into an equivalent Wheeler graph is equally hard as indexing a graph for linear time string search, but as far as we know, such equivalence result has not yet been established. Therefore the hardness of indexing graphs is largely still open.

In this paper we refute the existence of such a polynomial indexing scheme for graphs, under OVH. This contributes to a growing number of conditional lower bounds for offline string problem, such as the one for indexed *jumbled pattern matching* [6], conditioned on 3SUM-hardness, and the one for indexed *approximate pattern matching under κ differences* [15], conditioned on SETH.

Our result holds even for deterministic DAGs with labels from binary alphabet. By introducing a super-source connected to all source nodes and moving labels to incoming edges, such graphs can be interpreted as *acyclic non-deterministic finite automata* (acyclic NFA) whose only non-deterministic state is the start state. It follows that determinisation of such simple NFAs cannot be done in polynomial time unless OVH is false (and thus also unless SETH is false). This corollary complements the current picture of the exponential gap between NFAs and DFAs.

Table 1 and Figure 1 summarize the complexity landscape around offline SMLG.

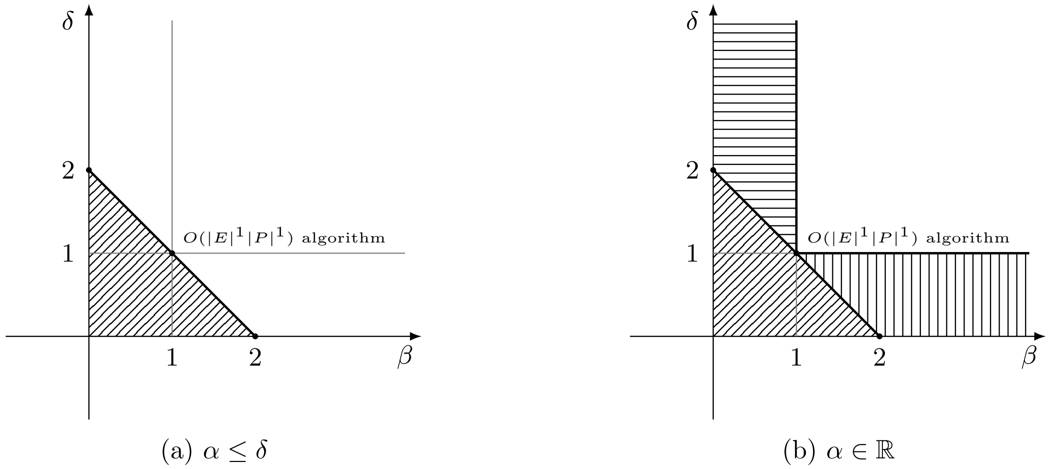
1.2 Results

In the Orthogonal Vectors (OV) problem we are given two sets $X, Y \subseteq \{0,1\}^d$ such that $|X| = |Y| = N$ and $d = \omega(\log N)$, and we need to decide whether there exists $x \in X$ and $y \in Y$ such that x and y are orthogonal, namely, $x \cdot y = 0$. OVH states that for any constant $\varepsilon > 0$, no algorithm can solve OV in time $O(N^{2-\varepsilon}\text{poly}(d))$. Notice that the better known *Strong Exponential Time Hypothesis* (SETH) [31] implies OVH [44], so all our lower bounds hold also under SETH.

Our results are obtained using a technique used for example in the field of dynamic algorithms, see e.g. [4, 2]. Recall the reduction from k-SAT to OV from [44]: the n variables of the formula ϕ are split into two groups of $n/2$ variables each, all partial $2^{n/2}$ Boolean assignments are generated for each group, and these induce two sets X and Y of size $N = 2^{n/2}$ each, such that OV returns 'yes' on X and Y if and only if ϕ is satisfiable. Suppose one could index X to support $O(M^{2-\varepsilon}\text{poly}(d))$-time queries for any set Y of M vectors, for some $\varepsilon > 0$. One now can adjust the splitting of the variables based on the hypothetical ε: the first part (corresponding to X) has $n\delta_\varepsilon$ variables, and the other part (corresponding to Y) has $n(1-\delta_\varepsilon)$ variables. We can choose a δ_ε depending on ε such that querying each vector in Y against the index on X takes overall time $O(2^{n(1-\gamma)})$, for some

Graph	Indexing time	Query time	Reference, Year
path	$O(\lvert E\rvert)$	$O(\lvert P\rvert)$	classical [18]
tree	$O(\lvert E\rvert)$	$O(\lvert P\rvert)$	[23], 2009
Wheeler graph	$O(\lvert E\rvert)$	$O(\lvert P\rvert)$	[43, 25], 2014
DAG	$O(\lvert E\rvert^{\alpha})$, $\alpha < 2$	$f(\lvert P\rvert)$ impossible under SIC	[12], 2013
arbitrary	$O(\lvert E\rvert^{\alpha})$, $\alpha \leq \delta$	$O(\lvert E\rvert^{\delta}\lvert P\rvert^{\beta})$, $\delta + \beta < 2$ impossible under OVH	[11], 2016
deterministic DAG	$O(\lvert E\rvert^{\alpha})$, $\alpha \leq \delta$	$O(\lvert E\rvert^{\delta}\lvert P\rvert^{\beta})$, $\delta + \beta < 2$ impossible under OVH	[20], 2019
deterministic DAG	$O(\lvert E\rvert^{\alpha})$, $\alpha \in \mathbb{R}$	$O(\lvert E\rvert^{\delta}\lvert P\rvert^{\beta})$, $\delta + \beta < 2$ impossible under OVH	This paper
arbitrary	$O(\lvert E\rvert^{\alpha})$, $\alpha \in \mathbb{R}$	$O(\lvert E\rvert^{\delta}\lvert P\rvert^{\beta})$, $\delta < 1$ or $\beta < 1$ impossible under OVH	This paper

Table 1: Upper bounds (first three rows) and conditional lower bounds for offline SMLG on a graph $G = (V, E)$ and a pattern P. On the fourth line, $f(\cdot)$ is an arbitrary function.



(a) $\alpha \leq \delta$

(b) $\alpha \in \mathbb{R}$

Figure 1: The dashed areas of the plots represent the forbidden values of δ and β for $O(|E|^{\delta}|P|^{\beta})$-time queries, under OVH. Figure 1a shows the lower bound that follows from the online case [11, 20], and holds for $\alpha \leq \delta$. Figure 1b depicts our lower bounds (tight, thanks to the online $O(|E||P|)$-time algorithm from [8]). In addition, these hold for any value of α.

$\gamma > 0$, contradicting SETH.

In this paper, instead of employing this technique inside the reduction for offline SMLG (as done in previous applications of this technique), we formalize the reason why it works through the notion of a *linear independent-component reduction (lic)*. Such a reduction allows to immediately argue that if a problem A is hard to index, and we have a *lic* reduction from A to B, then also B is hard to index (Lemma 1). Since OV is hard to index, it follows simply as a corollary that any problem to which OV reduces is hard to index. In order to get the best possible result for SMLG, we also show that a generalized version of OV is hard to index (Theorem 5). As such, we upgrade the idea of an "adjustable splitting" of the variables from a technique to a directly transferable result, once a *lic* reduction is shown to exist.

Examples of problems to which a *lic* reduction could be applied are those that arise from computing a distance between two elements. Popular examples are edit distance, dynamic time warping distance (DTWD), Frechet distance, longest common subsequence. All these problems have been shown to require quadratic time to be solved under OVH. The reductions proving these lower bound for DTWD [1] and Frechet distance [13] are indeed *lic* reductions, hence these problems automatically obtain a lower bound also for their offline version. More specifically, OVH implies that we cannot preprocess the first input of a DTWD or Frechet distance problem in polynomial time and provide sub-quadratic time queries for the second input.

On the other hand, the final sequences used in the reductions for edit distance [10] and longest common subsequence [1] present some dependencies within each other, thus they would need to be slightly tweaked to make the definition of *lic* reduction apply. These cross dependencies only concerns the size of the gadgets used in the reductions and not their structural properties, hence we are confident that the modifications needed to such gadgets require only a marginal effort.

To easily explain this idea and to better understand the utility of a *lic* reduction, let us consider edit distance. In a common offline variation of this problem, we are required to build a data structure for a long string T such that one can decide if a given query string P is within edit distance κ from a substring of T. It suffices to observe that, in the reduction of Backurs and Indyk [10] from OV to edit distance, this problem is utilized as an intermidiate step, and up to this point the reduction from OV is indeed a *lic* reduction (see Section 2.2). Hence, we immediately obtain the following result.

Theorem 1. *For any $\alpha > 0$, $\beta \geq 1$, and $\delta > 0$ such that $\beta + \delta < 2$, there is no algorithm preprocessing a string T in time $O(|E|^{\alpha})$, such that for any pattern string P we can find a substring of T at minimum edit distance with P, in time $O(|T|^{\delta}|P|^{\beta})$, unless OVH is false.*

This bound is tight because for $\delta + \beta = 2$ there is a matching online algorithm [34]. Theorem 1 also strenghtens the recent result of Cohen-Addad, Feuilloley and Starikovskaya [15], stating that an index built in polynomial time cannot support queries for approximate string matching in $O(|T|^{\delta})$ time, for any $\delta < 1$, unless SETH is false. However, the boundary condition is different, since in their case $\kappa = O(\log |T|)$, while in our case $\kappa = \Theta(|P|)$.

Our approach for the SMLG problem is similar. In Section 3 we revisit the reduction from [21] and observe that it is a *lic* reduction. As such, we can immediately obtain the following result.

Theorem 2. *For any $\alpha > 0$, $\beta \geq 1$, and $\delta > 0$ such that $\beta + \delta < 2$, there is no algorithm preprocessing a labeled graph $G = (V, E, \ell)$ in time $O(|E|^{\alpha})$ such that for any pattern string P we can solve the SMLG problem on G and P in time $O(|E|^{\delta}|P|^{\beta})$, unless OVH is false. This holds even if restricted to a binary alphabet, and to deterministic DAGs in which the sum of out-degree and in-degree of any node is at most three.*[1]

This lower bound is tight because for $\delta + \beta = 2$ there is a matching online algorithm [7]. However, this bound does not disprove a hypothetical polynomial indexing algorithm with query time $O(|E|^{\delta}|P|^{2})$, for some $0 < \delta < 1$. Since graphs in practical applications are much larger than the pattern, such an algorithm would be quite significant. However, when the graph is allowed to have cycles, we also show that this is impossible under OVH.

[1] We implicitly assumed here that the graph G is the part of the input on which to build the index, because it is the first input to SMLG. However, by exchanging G and P, it trivially holds that we also cannot polynomially index a pattern string P to support fast queries in the form of a labeled graph.

Theorem 3. *For any $\alpha > 0$, $\beta \geq 1$, and $0 < \delta < 1$, there is no algorithm preprocessing a labeled graph $G = (V, E, \ell)$ in time $O(|E|^{\alpha})$ such that for any pattern string P we can solve the* SMLG *problem on G and P in time $O(|E|^{\delta}|P|^{\beta})$, unless* OVH *is false.*

We obtain Theorem 3 by slightly modifying the reduction of [21] with the introduction of certain cycles, that are necessary to allow for query patterns of length longer than the graph size. We leave as open question whether the lower bound from Theorem 3 holds also for DAGs.

Open Problem 1. Do there exist $\alpha > 0$, $\beta \geq 1$, $0 < \delta < 1$, and an algorithm preprocessing a labeled (deterministic) DAG $G = (V, E, \ell)$ in time $O(|E|^{\alpha})$ such that for any pattern string P we can solve the SMLG problem on G and P in time $O(|E|^{\delta}|P|^{\beta})$?

2 Formalizing the technique

2.1 Linear independent-components reductions

All problems considered in this paper are such that their input is naturally partitioned in two. For a problem P, we will denote by $P_X \times P_Y$ the set of all possible inputs for P. For a particular input $(p_x, p_y) \in P_X \times P_Y$, we will denote by $|p_x|$ and $|p_y|$ the length of each of p_x and p_y, respectively. Intuitively, p_x represents what we want to build the index on, while p_y is what we want to query for. We start by formalizing the concept of *indexability*.

Definition 1 (Indexability)**.** Problem P is (I,Q)-*indexable* if for every $p_x \in P_X$ we can preprocess p_x in time $I(|p_x|)$ such that for every $p_y \in P_Y$ we can solve P on (p_x, p_y) in time $Q(|p_x|, |p_y|)$.

We further refine this notion into that of *polynomial indexability*, by specifying the degree of the polynomial costs of building the index and of performing the queries.

Definition 2 (Polynomial indexability)**.** Problem P is (α, δ, β)*-polynomially indexable with parameter k* if P is (I,Q)-indexable and $I(|p_x|) = O(k^{O(1)}|p_x|^{\alpha})$ and $Q(k^{O(1)}|p_x|, |p_y|) = O(|p_x|^{\delta}|p_y|^{\beta})$. If $k = O(1)$, then we say that P is (α, δ, β)*-polynomially indexable.*

The introduction of parameter k is needed to be consistent with OVH, since when proving a lower bound conditioned on OVH, the reduction is allowed to be polynomial in the vector dimension d. As we will see, we will set $k = d$.

We now introduce linear independent-components reductions, which we show below in Lemma 1 to maintain (α, δ, β)-polynomial indexability.

Definition 3 (*lic* reduction)**.** Problem A has a *linear independent-components (lic) reduction with parameter k* to problem B, indicated as $A \leq_{lic}^{k} B$, if the following two properties hold:

i) **Correctness**: There exists a reduction from A to B modeled by functions r_x, r_y and s. That is, for any input (a_x, a_y) for A, we have $r_x(a_x) = b_x$, $r_y(a_y) = b_y$, (b_x, b_y) is a valid input for B, and s solves A given the output $B(b_x, b_y)$ of an oracle to B, namely $s(B(r(a_x), r(a_y))) = A(a_x, a_y)$.

ii) **Parameterized linearity**: Functions r_x, r_y and s can be computed in linear time in the size of their input, multiplied by $k^{O(1)}$.

Lemma 1. *Given problems A and B and constants $\alpha > 0, \delta > 0, \beta \geq 1$, if $A \leq_{lic}^{k} B$ holds, and B is (α, δ, β)-polynomially indexable, then A is (α, δ, β)-polynomially indexable with parameter k.*

Proof. Let $a_x \in A_X$ be the first input of problem A. The linear independent-components reduction computes the first input of problem B as $b_x = r_x(a_x)$ in time $O(k^{O(1)}|a_x|)$. This means that $|b_x| = O(k^{O(1)}|a_x|)$, since the size of the data structure that we build with the reduction cannot be greater than the time spent for performing the reduction itself. Problem B is (α, δ, β)-polynomially indexable, hence we can build an index on b_x in time $O(|b_x|^\alpha)$ in such a way that we can perform queries for every b_y in time $O(|b_x|^\delta |b_y|^\beta)$. Now given any input a_y for A we can compute its corresponding $b_y = r_y(a_y)$ via the reduction in time $O(k^{O(1)}|a_y|)$ and answer a query for it using the index that we built on b_x. Again, notice that $|b_y| = O(k^{O(1)}|a_y|)$. The cost for such a query is $O(k^{O(1)}|a_y| + |b_x|^\delta |b_y|^\beta) = O(k^{O(1)}|a_y| + k^{O(1)}|a_x|^\delta |a_y|^\beta)$, which, since $\delta > 0$, is the same as $O(k^{O(1)}|a_x|^\delta |a_y|^\beta)$ when $\beta \geq 1$. Notice that the indexing time is $O(|b_x|^\alpha) = O(k^{O(1)}|a_x|^\alpha)$. Hence A is (α, δ, β)-polynomially indexable with parameter k, when $\beta \geq 1$. □

2.2 Conditional indexing lower bounds

We begin by stating, with our formalism, a known strengthening of the hardness of indexing reduction presented at the beginning of Section 1.2 (note that it also follows as a special case of Theorem 5 below).

Theorem 4 (Folklore). *If* OV *is (α, δ, β)-polynomially indexable with parameter d, and $\beta + \delta < 2$, then* OVH *fails.*

The value of a parameterized *lic* reduction can now be apprehended: once a parameterized *lic* reduction is shown to exist, the indexing lower bound follows directly.

Corollary 1. *Any problem P such that* OV $\leq^d_{lic}$ *P holds is not (α, δ, β)-polynomially indexable, for any $\alpha > 0$, $\beta \geq 1$, $\delta > 0$ with $\beta + \delta < 2$, unless* OVH *is false.*

Proof. Assume by contradiction that P is (α, δ, β)-polynomially indexable. Apply Lemma 1 to prove that OV is (α, δ, β)-polynomially indexable with parameter d, and $\beta + \delta < 2$; this contradicts Theorem 4. □

For a simple and concrete application of Corollary 1, consider the following problem, in which $d(S_1, S_2)$ denotes the edit distance between strings S_1 and S_2.

Problem 2 (PATTERN).

INPUT: Two strings T and P.

OUTPUT: $\min\limits_{S \text{ substring of } T} d(S, P)$.

Backurs and Indyk [10] reduce OV to PATTERN by constructing a string T based solely on the first input X to OV and a string P based solely on the second input Y to OV, such that if there are two orthogonal vectors then the answer to PATTERN on T and P is below a certain value, and if there are not, then the answer is equal to another specific value. Each of T and P can be constructed in time $O(d^{O(1)}N) = O(d^{O(1)}(dN))$. This is a *lic* reduction with parameter d. Directly applying Corollary 1, we obtain Theorem 1.

2.3 Indexing Generalized Orthogonal Vectors

Corollary 1 will suffice to prove Theorem 2. However, in order to prove that no query time $O(|E|^\delta |P|^\beta)$ is possible for any $\delta < 1$, we need a strengthening of Theorem 4. As such, we introduce the generalized (N, M)-*Orthogonal Vectors* problem, as follows:

Problem 3 ((N, M)-OV)**.**

Input: Two sets $X, Y \subseteq \{0,1\}^d$, such that $|X| = N$ and $|Y| = M$.

Output: *True* if and only if there exists $(x, y) \in X \times Y$ such that $x \cdot y = 0$.

The theorem below is the desired generalization of Theorem 4, since it implies, for example, that we cannot have $O(N^{1/2}M^2)$-time queries after polynomial-time indexing. To the best of our efforts, we could not find a proof of this result in the literature, and hence we give one here. It is based on the same idea of an "adjustable splitting" into subvectors, a part of which is indexed, while the other part is queried. However, some technical subtleties arise from the combination of all parameters α, δ, β.

Theorem 5. *If (N, M)-OV is (α, δ, β)-polynomially indexable with parameter d, and either $\delta < 1$ or $\beta < 1$, then OVH fails. That is, under OVH, we cannot support $O(N^\delta M^\beta)$-time queries for (N, M)-OV, for either $\delta < 1$ or $\beta < 1$, even after polynomial-time preprocessing.*

Proof. Let X and Y be the input for OV and assume that their length is n. Our strategy is to split this instance of OV into many (N, M)-OV sub-problems and show that a too efficient indexing scheme for (N, M)-OV applied to such sub-problems would lead to an online algorithm for OV running in sub-quadratic time, hence contradicting OVH. The key is to adjust the size of such (N, M)-OV sub-problems to fit our needs. Let us begin by partitioning set X into subsets of N vectors each, and set Y into subsets of M vectors each, as shown in Figure 2. The instances of (N, M)-OV sub-problems that we want to consider are all those pairs of vector sets (X_i, Y_j) in which X_i is a subset of X and Y_j is a subset of Y. Solving all the (X_i, Y_j) instances solves the original problem.[2] Now we proceed to index sub-sets X_i and to analyze how the time complexity of the original problem looks like when expressed in terms of the (N, M)-OV sub-problems. Later we show how we can obtain a sub-quadratic time algorithm for OV by choosing specific values for N and M.

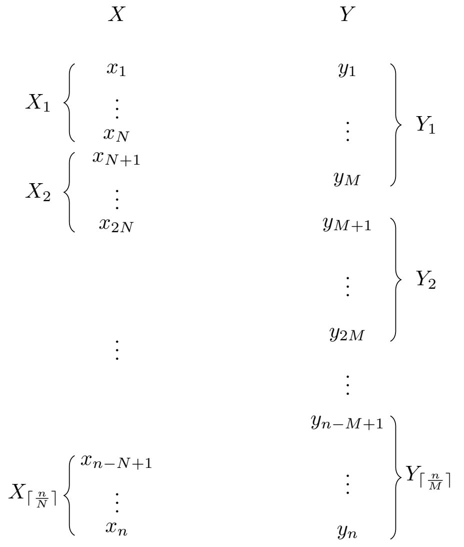


Figure 2: Two sets of n vectors and their partitioning into sub-sets of X for indexing, and sub-sets of Y for querying.

[2] The idea of splitting the two sets into smaller groups was also used in [3] to obtain a fast randomized algorithm for OV, based on the polynomial method, and therein the groups always had equal size.

Since we are assuming that (N, M)-OV is (α, δ, β)-polynomially indexable with parameter d, we can build index $Idx(X_i)$ for subset X_i of N vectors in time $O(d^{O(1)}(dN)^{\alpha})$, and additionally we can answer a query for any subset Y_j of M vectors using index $Idx(X_i)$ in time $O(d^{O(1)}(dN)^{\delta}(dM)^{\beta})$. Hence, given index $Idx(X_i)$, we can solve sub-problems (X_i, Y_j) for a fixed i and j $(1 \leq j \leq \lceil \frac{n}{M} \rceil)$ by performing $\lceil \frac{n}{M} \rceil$ queries, one for each subset Y_j of Y. Repeating this process for all X_i covers all possible pairs (X_i, Y_j), and since we have $\lceil \frac{n}{N} \rceil \lceil \frac{n}{M} \rceil$ such pairs, the total cost for solving OV is:

$$O\left(d^{O(1)}(dN)^{\alpha}\frac{n}{N} + d^{O(1)}(dN)^{\delta}(dM)^{\beta}\frac{n}{N}\frac{n}{M}\right) = O\left(d^{O(1)}\left(N^{\alpha-1}n + N^{\delta-1}M^{\beta-1}n^2\right)\right). \quad (1)$$

In order to have a sub-quadratic-time algorithm for OV we need both of the terms of the sum above to be sub-quadratic. Namely, our time complexity should be $O\left(d^{O(1)}\left(n^{2-\varepsilon'} + n^{2-\varepsilon}\right)\right)$, for some ε, $\varepsilon' > 0$. Notice that in order to prove OVH to be wrong it is enough to find one specific value for ε and one for ε' such that the following two conditions hold:

$$\begin{aligned}&\text{(a)}: \ N^{\alpha-1}n = O(n^{2-\varepsilon'})\\ &\text{(b)}: \ N^{\delta-1}M^{\beta-1}n^2 = O(n^{2-\varepsilon})\end{aligned}$$

As a first observation, notice that we need also to enforce $1 \leq N \leq n$ and $1 \leq M \leq n$. This is because every X_i and every Y_j must contain at least one vector in order to be an instance of (N, M)-OV, and trivially their size must not exceed the size n of the original OV instance. Moreover, N and M must also be integers. This last requirement might cause some complications during our analysis, and due to this reason we will take advantage of a useful trick. We introduce new variables $\tilde{N}$ and $\tilde{M}$ so that we can make them assume real values. Our actual N and M would be the ceiling of $\tilde{N}$ and $\tilde{M}$. Putting all together, we want that for every $n \in \mathbb{N}, \alpha, \delta, \beta > 0$ such that either $\delta < 1$ or $\beta < 1$ there exists $\varepsilon > 0, \varepsilon' > 0, N, M, \tilde{N}, \tilde{M}$ such that:

$$\begin{aligned}&\text{(a) } N^{\alpha-1}n = O(n^{2-\varepsilon'})\\ &\text{(b) } N^{\delta-1}M^{\beta-1}n^2 = O(n^{2-\varepsilon})\\ &(\tilde{\text{a}}) \ \tilde{N}^{\alpha-1}n = n^{2-\varepsilon'}\\ &(\tilde{\text{b}}) \ \tilde{N}^{\delta-1}\tilde{M}^{\beta-1}n^2 = n^{2-\varepsilon}\\ &\text{(c) } N = \lceil \tilde{N} \rceil, \ M = \lceil \tilde{M} \rceil\\ &\text{(d) } 1 \leq \tilde{N} \leq n, \ 1 \leq \tilde{M} \leq n\end{aligned}$$

Notice that forcing $1 \leq \tilde{N} \leq n$ also ensures $1 \leq N \leq n$, since we are taking the ceiling $N = \lceil \tilde{N} \rceil$. The same holds for $\tilde{M}$ and M.

We start our case analysis by identifying two cases for parameter α, namely $\alpha \neq 1$ and $\alpha = 1$. These are eventually broken down into specific sub-cases for parameters δ and β. The strategy is to prove that if conditions (ã), ($\tilde{\text{b}}$), (c), (d), (e) hold, then also conditions (a) and (b) hold. For simplicity, we report here only the most interesting cases in which $\alpha \neq 0$, $\delta \neq 1$ and $\beta \neq 1$. The complete analysis of the remaining cases can be found in Appendix B.

Case 1: $\alpha \neq 1$. In this case we obtain the following constraint on $\tilde{N}$ from condition (ã):

$$\tilde{N}^{\alpha-1}n = n^{2-\varepsilon'} \Leftrightarrow \tilde{N} = n^{\frac{1-\varepsilon'}{\alpha-1}}. \quad (2)$$

Now, given ε', we can compute $\tilde{N}$ using this equation so that we satisfy condition (ã). In doing so we need to make sure that condition (d) is also respected. To this end, we need to check that ε'

satisfies $0 \leq \frac{1-\varepsilon'}{\alpha-1} \leq 1$. Let us start with the left inequality.

$$\begin{aligned}\frac{1-\varepsilon'}{\alpha-1} \geq 0 &\Leftrightarrow (1-\varepsilon' \geq 0 \text{ and } \alpha - 1 > 0) \text{ or } (1-\varepsilon' \leq 0 \text{ and } \alpha - 1 < 0)\\ &\Leftrightarrow (\varepsilon' \leq 1 \text{ and } \alpha > 1) \text{ or } (\varepsilon' \geq 1 \text{ and } \alpha < 1) \qquad (3)\end{aligned}$$

For the right inequality, we first handle the case in which $\varepsilon' \leq 1$ and $\alpha > 1$ and we combine it with the constraint $\frac{1-\varepsilon'}{\alpha-1} \leq 1$. Since $\alpha > 1$ then $\alpha - 1 > 0$ and we have that $\frac{1-\varepsilon'}{\alpha-1} \leq 1 \Leftrightarrow \varepsilon' \geq 2-\alpha$. Hence, the final constraint for ε' is $2-\alpha \leq \varepsilon' \leq 1$, and we know that there exists valid values for ε' since $\alpha > 1 \Rightarrow 2-\alpha < 1$.

Now we take into account the other case, namely $\varepsilon' \geq 1$ and $\alpha < 1$. We find ourselves in a symmetric situation in which $\alpha < 1$ implies $\alpha - 1 < 0$ which leads to $\frac{1-\varepsilon'}{\alpha-1} \leq 1 \Leftrightarrow \varepsilon' \leq 2-\alpha$. Putting all together we have $1 \leq \varepsilon' \leq 2-\alpha$, and the existence of valid values for ε' is guaranteed by the fact that $\alpha < 1 \Rightarrow 2-\alpha > 1$.

So far we analyzed conditions (ã) and (d), for $\tilde{N}$. To analyze the other conditions, we need to consider three sub-cases. Here we present the more challenging one, which is in turn split into two more sub-cases. The reader can find the others in Appendix B.

Case 1.1: $\delta \neq 1$ and $\beta \neq 1$. Now condition ($\tilde{\text{b}}$) yields the following:

$$\tilde{N}^{\delta-1}\tilde{M}^{\beta-1}n^2 = n^{2-\varepsilon} \Leftrightarrow \tilde{M} = \tilde{N}^{\frac{1-\delta}{\beta-1}} n^{\frac{\varepsilon}{1-\beta}}. \qquad (4)$$

We apply the substitution $\tilde{N} = n^{\frac{1-\varepsilon'}{\alpha-1}}$ that we obtained from equation (2). Hence:

$$\tilde{M} = n^{\frac{1-\varepsilon'}{\alpha-1}\frac{1-\delta}{\beta-1}} n^{\frac{\varepsilon}{1-\beta}} = n^{\frac{1-\varepsilon'}{\alpha-1}\frac{1-\delta}{\beta-1}-\frac{\varepsilon}{\beta-1}}.$$

We apply condition (d) obtaining the following constraint:

$$0 \leq \frac{1-\varepsilon'}{\alpha-1}\frac{1-\delta}{\beta-1} - \frac{\varepsilon}{\beta-1} \leq 1 \qquad (5)$$

Here we face two more sub-cases.

Case 1.1.1: $\beta - 1 < 0 \Leftrightarrow \beta < 1$. We extract the constraint on ε from the two inequalities in (5) above. We start by analysing the left inequality.

$$\begin{aligned}\frac{1-\varepsilon'}{\alpha-1}\frac{1-\delta}{\beta-1} - \frac{\varepsilon}{\beta-1} &\geq 0\\ \varepsilon &\geq \frac{1-\varepsilon'}{\alpha-1}(1-\delta).\end{aligned}$$

From the second inequality instead we get:

$$\begin{aligned}\frac{1-\varepsilon'}{\alpha-1}\frac{1-\delta}{\beta-1} - \frac{\varepsilon}{\beta-1} &\leq 1\\ \varepsilon - \frac{1-\varepsilon'}{\alpha-1}(1-\delta) &\leq 1-\beta\\ \varepsilon &\leq 1-\beta+\frac{1-\varepsilon'}{\alpha-1}(1-\delta)\end{aligned}$$

Since $\varepsilon > 0$, we need to be sure that the right term of this last inequality is strictly greater than 0. Since $1-\beta > 0$ and $\frac{1-\varepsilon'}{\alpha-1} > 0$ (from (3)), the interesting case is when $1-\delta < 0$. Notice that

$\frac{1-\varepsilon'}{\alpha-1} \to 0$ as $\varepsilon' \to 1$. This means that we can always choose ε' as close to 1 as needed to make $1-\beta+\frac{1-\varepsilon'}{\alpha-1}(1-\delta) > 0$ hold.

At this point we have proved conditions (ã), ($\tilde{\text{b}}$) and (d), thus we are left to show that conditions (a) and (b) also hold. To this end, let us choose ε, ε', $\tilde{N}$ and $\tilde{M}$ in such a way that conditions (ã), ($\tilde{\text{b}}$) and (d) are verified. Then we choose $N = \lceil N \rceil$ and $M = \lceil M \rceil$ so that condition (c) is verified. We analyse in depth condition (b) since it is more complicated; condition (a) can be proven applying the same technique. We first remark the following property:

Fact 1. $\forall n, a, b \in \mathbb{R}$. $\lceil n^a \rceil^b = O(n^{ab})$.

Proof. For $b = 0$ the statement is trivially true. If $b > 0$, we have $\lceil n^a \rceil^b \leq (n^a + 1)^b = O(n^{ab})$. If $b < 0$, we have $\lceil n^a \rceil^b \leq (n^a - 1)^b = O(n^{ab})$. □

Now we can show that

$$\begin{aligned} N^{\delta-1} M^{\beta-1} n^2 &= \lceil n^{\frac{1-\varepsilon'}{\alpha-1}} \rceil^{\delta-1} \lceil n^{\frac{1-\varepsilon'}{\alpha-1}\frac{1-\delta}{\beta-1} - \frac{\varepsilon}{\beta-1}} \rceil^{\beta-1} n^2 \\ &= O\left(n^{-\frac{1-\varepsilon'}{\alpha-1}(1-\delta)} n^{\frac{1-\varepsilon'}{\alpha-1}(1-\delta)-\varepsilon} n^2 \right) \\ &= O(n^{2-\varepsilon}) \end{aligned}$$

where the first step is justified by Fact 1. We conclude that both conditions (a) and (b) hold.

Case 1.1.2: $\beta - 1 > 0 \Leftrightarrow \beta > 1$. This case is symmetric to the previous one and implies that we are in the situation $\delta < 1$. When extracting the constraints on ε, we will have the same inequalities but with the opposite direction. Indeed, we multiply by $\beta - 1$ which now has the opposite sign.

$$1 - \beta + \frac{1-\varepsilon'}{\alpha-1}(1-\delta) \leq \varepsilon \leq \frac{1-\varepsilon'}{\alpha-1}(1-\delta)$$

Given that $\varepsilon > 0$, we need to verify to have room to choose such an ε, that is $\frac{1-\varepsilon'}{\alpha-1}(1-\delta) > 0$. We know that the first factor of this multiplication is between 0 and 1. Hence, we can always choose an ε' such that $\frac{1-\varepsilon'}{\alpha-1} > 0$. Moreover, in this sub-case we have $\delta < 1$ which ensures that also $1 - \delta$ is strictly positive. Hence, the quantity $\frac{1-\varepsilon'}{\alpha-1}(1-\delta)$ is strictly positive, which means that there always exists an ε such that condition (d) holds. Assuming condition (c), conditions (a) and (b) can be proved to hold in the same manner as in the previous sub-case.

In conclusion, we can say that depending on α, δ and β we find ourselves into one of the listed cases. We showed that in each one of those we can always find values for ε and ε' such that there exists integer values for N and M that provide an algorithm for OV running in time $O(n^{2-\varepsilon}+n^{2-\varepsilon'})$, proving OVH to be false. □

Corollary 2. *Any problem P such that (N, M)-OV $\leq^d_{lic}$ P holds is not (α, δ, β)-polynomially indexable, for any $\alpha > 0$, $\beta \geq 1$, $0 < \delta < 1$, unless OVH is false.*

3 Indexing Labeled Graphs for String Matching

Recall the following conditional lower bound for SMLG from Equi et al. [21].

Theorem 6 ([21])**.** *For any $\varepsilon > 0$, SMLG on labeled deterministic DAGs cannot be solved in either $O(|E|^{1-\varepsilon}\,|P|)$ or $O(|E|\,|P|^{1-\varepsilon})$ time unless OVH fails. This holds even if restricted to a binary alphabet, and to DAGs in which the sum of out-degree and in-degree of any node is at most three.*

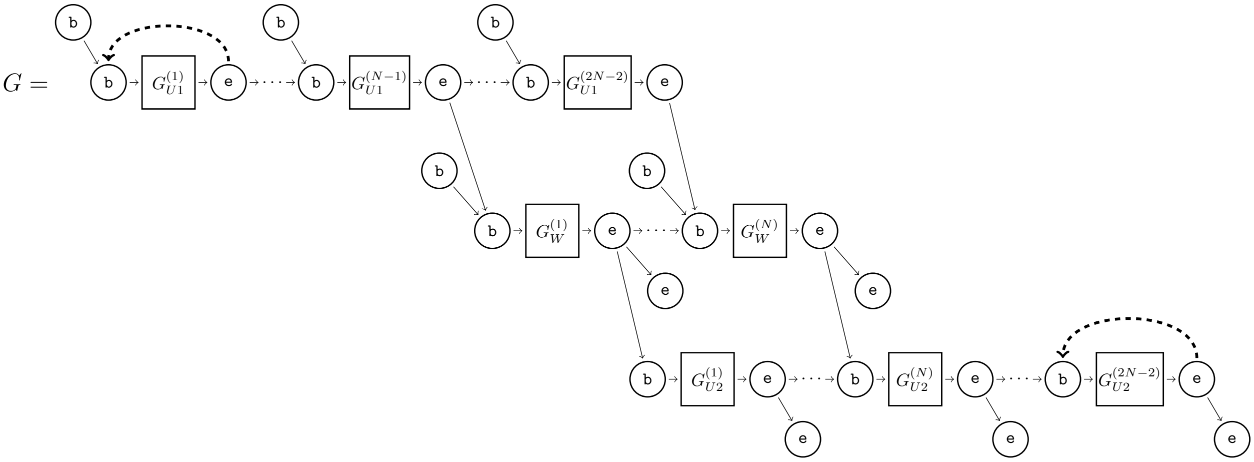


Figure 3: Non-deterministic graph G. The dashed thick edges are not present in the acyclic graph from [21], and must be added to handle (N, M)-OV instances with $M > N$.

Given an OV instance with sets X and Y, the reduction from [21] builds a graph G using solely X, and a pattern P using solely Y, both in linear time $O(dN)$, such that P has a match in G if and only if there exists a pair of orthogonal vectors.[3] This shows that the two conditions of the linear independent-components reduction property hold, thus $\mathsf{OV} \leq_{lic}^{d} \mathsf{SMLG}$. Directly applying Corollary 1, we obtain Theorem 2.

Next, we show that constraint $\beta + \delta < 2$ can be dropped from Theorem 2 when we are indexing non-deterministic graphs with cycles. The idea is that if we allow (N, M)-OV instances with $M > N$, then the reduction from [21] no longer holds, because the pattern P is too large to fit inside the DAG G. As such, we need to make a minor adjustment to G. For this, we must give some additional details of that reduction. For our purposes, it is enough to explain the construction of a non-deterministic graph from [21, Section 2.3].

Pattern P is over the alphabet $\Sigma = \{\mathtt{b}, \mathtt{e}, \mathtt{0}, \mathtt{1}\}$, has length $|P| = O(dM)$, and can be built in $O(dM)$ time from the second set of vectors $Y = \{y_1, \dots, y_M\}$. Namely, we define

$$P = \mathtt{bb}P_{y_1}\mathtt{e}\,\mathtt{b}P_{y_2}\mathtt{e}\dots\mathtt{b}P_{y_M}\mathtt{ee}$$

where P_{y_i} is a string of length d that is associated with each $y_i \in Y$, for $1 \leq i \leq M$. The h-th symbol of P_{y_i} is either $\mathtt{0}$ or $\mathtt{1}$, for each $h \in \{1, \dots, d\}$, such that $P_{y_i}[h] = \mathtt{1}$ if and only if $x_i[h] = 1$.

Starting from the first set of vectors X, we define the directed graph $G_W = (V_W, E_W, L_W)$, which can be built in $O(dN)$ time and consists of N connected components $G_W^{(j)}$, one for each vector $x_j \in X$. Component $G_W^{(j)}$ can be constructed so that it contains an occurrence of a subpattern P_{y_i} if and only if $x_j \cdot y_i = 0$. In addition, we need a universal gadget $G_U = (V_U, E_U, L_U)$ of $2N - 2$ components $G_{U1}^{(k)}$, where each component can match any of the subpatterns P_{y_i}. We actually need two copies G_{U1} and G_{U2} of such universal gadgets, a “top” one, and a “bottom” one, respectively. All the gadgets are connected as indicated in Figure 3 and the resulting graph G has total size $O(dN)$.

The intuition is that a prefix of P is handled by the “top” universal gadgets G_{U1}, a possible matching a subpattern P_{y_i} of P by one of the “middle” gadgets $G_W^{(j)}$, and a suffix of P by the “bottom” universal gadgets, because P has a $\mathtt{bb}$ prefix and an $\mathtt{ee}$ suffix. As mentioned above, by

[3] Notice that [21] originally built P based on X, and G based on Y. Since it is immaterial for correctness, and in order to keep in line with the notation in this paper, we assumed the opposite here.

doing this we cannot accommodate (N, M)-OV instances with $M > N$. However, we can easily fix this by adding a cycle in each of the "top" and "bottom" universal gadgets, so that a longer pattern will match a universal gadget in this cycle as many times needed to fit inside the graph. More precisely, we can add an edge from the e-node to the right of $G_{U1}^{(1)}$ back to the b-node to the left of $G_{U1}^{(1)}$, and likewise from the e-node to the right of $G_{U2}^{(2N-2)}$ back to the b-node to the left of $G_{U2}^{(2N-2)}$ (see Figure 3). It can easily be checked that it still holds that P has a match in the resulting graph G if and only if there are two orthogonal vectors, no matter the relationship between N and M. Applying Corollary 2, we obtain Theorem 3.

References

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78, 2015.

[2] Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 25–36. IEEE Computer Society, 2017. URL: `https://doi.org/10.1109/FOCS.2017.12`, `doi:10.1109/FOCS.2017.12`.

[3] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 218–230, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=2722129.2722146`.

[4] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443. IEEE Computer Society, 2014. URL: `https://doi.org/10.1109/FOCS.2014.53`, `doi:10.1109/FOCS.2014.53`.

[5] Jarno Alanko, Giovanna D'Agostino, Alberto Policriti, and Nicola Prezza. Regular languages meet prefix sorting. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 911–930. SIAM, 2020. URL: `https://doi.org/10.1137/1.9781611975994.55`, `doi:10.1137/1.9781611975994.55`.

[6] Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2014. URL: `https://doi.org/10.1007/978-3-662-43948-7_10`, `doi:10.1007/978-3-662-43948-7\_10`.

[7] Amihood Amir, Moshe Lewenstein, and Noa Lewenstein. Pattern matching in hypertext. In *WADS'97, Halifax, LNCS 1272*, pages 160–173, 1997.

[8] Amihood Amir, Moshe Lewenstein, and Noa Lewenstein. Pattern matching in hypertext. *J. Algorithms*, 35(1):82–99, 2000.

[9] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39, February 2008. URL: `http://doi.acm.org/10.1145/1322432.1322433`, `doi:10.1145/1322432.1322433`.

[10] Arturs Backurs and Piotr Indyk. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (Unless SETH is False). In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 51–58, New York, NY, USA, 2015. ACM. URL: `http://doi.acm.org/10.1145/2746539.2746612`, `doi:10.1145/2746539.2746612`.

[11] Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 457–466, 2016.

[12] Philip Bille. Personal Communication at Dagstuhl Seminar on Indexes and Computation over Compressed Structured Data, June 2013.

[13] Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless seth fails. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 661–670. IEEE, 2014.

[14] M. Burrows and D. Wheeler. A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, 1994.

[15] Vincent Cohen-Addad, Laurent Feuilloley, and Tatiana Starikovskaya. Lower bounds for text indexing with mismatches and differences. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1146–1164. SIAM, 2019. URL: `https://doi.org/10.1137/1.9781611975482.70`, `doi:10.1137/1.9781611975482.70`.

[16] The Computational Pan-Genomics Consortium. Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, 19(1):118–135, 2018. URL: `http://dx.doi.org/10.1093/bib/bbw089`, `arXiv:/oup/backfile/content_public/journal/bib/19/1/10.1093_bib_bbw089/5/bbw089.pdf`, `doi:10.1093/bib/bbw089`.

[17] Alessio Conte, Gaspare Ferraro, Roberto Grossi, Andrea Marino, Kunihiko Sadakane, and Takeaki Uno. Node Similarity with q -Grams for Real-World Labeled Networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1282–1291, 2018. URL: `https://doi.org/10.1145/3219819.3220085`, `doi:10.1145/3219819.3220085`.

[18] Maxime Crochemore and Wojciech Rytter. *Jewels of stringology.* World Scientific, 2002. URL: `https://doi.org/10.1142/4838`, `doi:10.1142/4838`.

[19] Eggertsson Hannes P, Jonsson Hakon, Kristmundsdottir Snaedis, Hjartarson Eirikur, Kehr Birte, Masson Gisli, Zink Florian, Hjorleifsson Kristjan E, Jonasdottir Aslaug, Jonasdottir Adalbjorg, Jonsdottir Ingileif, Gudbjartsson Daniel F, Melsted Pall, Stefansson Kari, and Halldorsson Bjarni V. Graphtyper enables population-scale genotyping using pangenome graphs. *Nature Genetics*, 49(11):1654–1660, 2017. `doi:https://doi.org/10.1038/ng.3964`.

[20] Massimo Equi. Pattern matching in labeled graphs. Master's thesis, University of Pisa, Italy, 2018. URL: `https://etd.adm.unipi.it/theses/available/etd-09102018-185610/unrestricted/MasterThesis_MassimoEqui.pdf`.

[21] Massimo Equi, Roberto Grossi, Veli Mäkinen, and Alexandru I. Tomescu. On the complexity of string matching for graphs. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, volume 132 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. URL: `https://doi.org/10.4230/LIPIcs.ICALP.2019.55`, `doi:10.4230/LIPIcs.ICALP.2019.55`.

[22] P. Ferragina and G. Manzini. Indexing compressed texts. *Journal of the ACM*, 52(4):552–581, 2005.

[23] Paolo Ferragina, Fabrizio Luccio, Giovanni Manzini, and S. Muthukrishnan. Compressing and indexing labeled trees, with applications. *J. ACM*, 57(1):4:1–4:33, 2009.

[24] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 1433–1445, 2018. URL: `https://doi.org/10.1145/3183713.3190657`, `doi:10.1145/3183713.3190657`.

[25] Travis Gagie, Giovanni Manzini, and Jouni Sirén. Wheeler graphs: A framework for BWT-based data structures. *Theor. Comput. Sci.*, 698:67–78, 2017. URL: `https://doi.org/10.1016/j.tcs.2017.06.016`, `doi:10.1016/j.tcs.2017.06.016`.

[26] Garrison Erik, Sirén Jouni, Novak Adam M, Hickey Glenn, Eizenga Jordan M, Dawson Eric T, Jones William, Garg Shilpa, Markello Charles, Lin Michael F, Paten Benedict, and Durbin Richard. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature Biotechnology*, 36:875, aug 2018. URL: `https://www.nature.com/articles/nbt.4227#supplementary-information`, `doi:http://dx.doi.org/10.1038/nbt.422710.1038/nbt.4227`.

[27] Daniel Gibney and Sharma V. Thankachan. On the hardness and inapproximability of recognizing wheeler graphs. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany.*, volume 144 of *LIPIcs*, pages 51:1–51:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: `https://doi.org/10.4230/LIPIcs.ESA.2019.51`, `doi:10.4230/LIPIcs.ESA.2019.51`.

[28] Isaac Goldstein, Moshe Lewenstein, and Ely Porat. On the hardness of set disjointness and set intersection with bounded universe. In Pinyan Lu and Guochuan Zhang, editors, *30th International Symposium on Algorithms and Computation, ISAAC 2019, December 8-11, 2019, Shanghai University of Finance and Economics, Shanghai, China*, volume 149 of *LIPIcs*, pages 7:1–7:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: `https://doi.org/10.4230/LIPIcs.ISAAC.2019.7`, `doi:10.4230/LIPIcs.ISAAC.2019.7`.

[29] R. Grossi and J. Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. *SIAM Journal on Computing*, 35(2):378–407, 2006.

[30] Shohei Hido and Hisashi Kashima. A linear-time graph kernel. In Wei Wang 0010, Hillol Kargupta, Sanjay Ranka, Philip S. Yu, and Xindong Wu, editors, *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, pages 179–188. IEEE Computer Society, 2009.

[31] Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001. URL: `http://www.sciencedirect.com/science/article/pii/S0022000000917276`, `doi:https://doi.org/10.1006/jcss.2000.1727`.

[32] Kim Daehwan, Paggi Joseph M., Park Chanhee, Bennett Christopher, and Salzberg Steven L. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nature Biotechnology*, 37(8):907–915, 2019. `doi:https://doi.org/10.1038/s41587-019-0201-4`.

[33] Veli Mäkinen, Alexandru I. Tomescu, Anna Kuosmanen, Topi Paavilainen, Travis Gagie, and Rayan Chikhi. Sparse dynamic programming on DAGs with small width. *ACM Trans. Algorithms*, 15(2):29:1–29:21, 2019.

[34] 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. URL: `http://www.sciencedirect.com/science/article/pii/0022000080900021`, `doi:10.1016/0022-0000(80)90002-1`.

[35] Gonzalo Navarro and Veli Mäkinen. Compressed full-text indexes. *ACM Comput. Surv.*, 39(1):2, 2007. URL: `https://doi.org/10.1145/1216370.1216372`, `doi:10.1145/1216370.1216372`.

[36] Mihai Patrascu and Liam Roditty. Distance Oracles beyond the Thorup-Zwick Bound. *SIAM J. Comput.*, 43(1):300–311, 2014. URL: `https://doi.org/10.1137/11084128X`, `doi:10.1137/11084128X`.

[37] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. World Wide Web Consortium, Recommendation REC-rdf-sparql-query-20080115, January 2008.

[38] Mikko Rautiainen, Veli Mäkinen, and Tobias Marschall. Bit-parallel sequence-to-graph alignment. *Bioinformatics*, 35(19):3599–3607, 2019. URL: `https://doi.org/10.1093/bioinformatics/btz162`, `doi:10.1093/bioinformatics/btz162`.

[39] Mikko Rautiainen and Tobias Marschall. GraphAligner: Rapid and Versatile Sequence-to-Graph Alignment. *bioRxiv*, 2019. URL: `https://www.biorxiv.org/content/early/2019/10/21/810812`, `arXiv:https://www.biorxiv.org/content/early/2019/10/21/810812.full.pdf`, `doi:10.1101/810812`.

[40] Marko A. Rodriguez. The gremlin graph traversal machine and language (invited talk). In *Proceedings of the 15th Symposium on Database Programming Languages, Pittsburgh, PA, USA, October 25-30, 2015*, pages 1–10, 2015. URL: `https://doi.org/10.1145/2815072.2815073`, `doi:10.1145/2815072.2815073`.

[41] Korbinian Schneeberger, Jörg Hagmann, Stephan Ossowski, Norman Warthmann, Sandra Gesing, Oliver Kohlbacher, and Detlef Weigel. Simultaneous alignment of short reads against multiple genomes. *Genome Biology*, 10:R98, 2009.

[42] Jouni Sirén. Indexing variation graphs. In Sándor P. Fekete and Vijaya Ramachandran, editors, *Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2017, Barcelona, Spain, Hotel Porta Fira, January 17-18, 2017*, pages 13–27. SIAM, 2017. URL: `https://doi.org/10.1137/1.9781611974768.2`, `doi:10.1137/1.9781611974768.2`.

[43] Jouni Sirén, Niko Välimäki, and Veli Mäkinen. Indexing graphs for path queries with applications in genome research. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 11(2):375–388, March 2014. URL: `http://dx.doi.org/10.1109/TCBB.2013.2297101`, `doi:10.1109/TCBB.2013.2297101`.

[44] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. URL: `https://doi.org/10.1016/j.tcs.2005.09.023`, `doi:10.1016/j.tcs.2005.09.023`.

A Connection to SIC

Given sets $S^1, S^2, \ldots, S^n \subseteq [1..u]$, where $u = \log^c n$ for sufficiently large c, the *Set Intersection Conjecture (SIC)* [36] is that there is no index of size $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$ to answer in constant time if two sets S^i and S^j intersect or not (i.e. there is no improvement over the table of all precomputed solutions). The reduction of [12] is as follows: build a simple DAG with one copy of the sets as source nodes and another copy as sink nodes. Then add nodes in between corresponding to the elements of the sets. Connect source node corresponding to S^i to all nodes corresponding to elements $v \in S^i$, and all nodes corresponding to $v \in S^i$ to the sink corresponding to S^i, for all i. Label sources and sinks with their set identifier, and nodes in between with some common letter, say $\mathtt{A}$. Since the graph size is $O(n \log^c n)$, a truly sub-quadratic size index supporting string queries of the form $P = i\mathtt{A}j$ even, say, in exponential time in $|P|$ would prove SIC false. Modifying the relationship between universe size u and number of sets n gives rise to several refined lower bounds for the tradeoff betweeen index construction and query time [28], which directly transfer to the graph indexing problem through the simple connection stated above.

B Missing cases of the proof of Theorem 5

Case 2: $\alpha = 1$. Condition $(\tilde{a})$ simply becomes $n = n^{2-\varepsilon'}$, which is verified for $\varepsilon' = 1$. We now split the analysis of condition $(\tilde{b})$ into two sub-cases.

Case 2.1: $\delta < 1$ and no constraint on β. We can rewrite condition $(\tilde{b})$ as:

$$\tilde{N} = \tilde{M}^{\frac{1-\beta}{\delta-1}} n^{\frac{\varepsilon}{1-\delta}}$$

since $\delta < 1$ guarantees $\delta - 1 \neq 0$. If we choose $\tilde{M} = 1$ we respect condition (d) and we obtain $\tilde{N} = n^{\frac{\varepsilon}{1-\delta}}$ for any value of β. Hence, we can first choose a value for ε and later use this equation to obtain the right value for $\tilde{N}$ that will satisfy condition $(\tilde{b})$. Nevertheless, we cannot just pick any value for ε. Indeed, we need to guarantee also that condition (d) is holding. This can be achieved by verifying that $0 \leq \frac{\varepsilon}{1-\delta} \leq 1$. Since $\delta < 1$ and $\varepsilon > 0$ we know that $\frac{\varepsilon}{1-\delta} > 0$. Moreover, $\frac{\varepsilon}{1-\delta} \leq 1 \Leftrightarrow \varepsilon \leq 1 - \delta$, which means that any ε such that $0 < \varepsilon \leq 1 - \delta$ satisfies condition (d). We know that there exists such an ε since $1 - \delta > 0$.

We are now left to prove that conditions (a) and (b) hold. We proceed as in case 1.1.1 by assuming condition (c) and proving conditions (a) and (b). Condition (a) is easily verified since $\alpha = 1$. Since $N = \lceil \tilde{N} \rceil = \lceil n^{\frac{\varepsilon}{1-\delta}} \rceil$ and $M = \lceil \tilde{M} \rceil = 1$, and noticing that $\delta - 1 < 0$, we can analyse condition (b) as follows.

$$\begin{aligned} N^{\delta-1} M^{\beta-1} n^2 &= \lceil n^{\frac{\varepsilon}{1-\delta}} \rceil^{\delta-1} n^2 \\ &\leq \left(n^{\frac{\varepsilon}{1-\delta}} - 1 \right)^{\delta-1} \cdot n^2 \\ &= O(n^{\frac{\varepsilon}{1-\delta}\delta-1} n^2) \\ &= O(n^{2-\varepsilon}). \end{aligned}$$

Hence, condition (b) is verified and so all the conditions hold.

Case 2.2: $\beta < 1$ and no constraint on δ. This case is symmetric to the previous one. Indeed, we now rewrite condition $(\tilde{b})$ as:

$$\tilde{M} = \tilde{N}^{\frac{1-\delta}{\beta-1}} n^{\frac{\varepsilon}{1-\beta}}$$

where $\beta < 1$ gives $\beta - 1 \neq 0$. This time we choose $\tilde{N} = 1$, from which we obtain $\tilde{M} = n^{\frac{\varepsilon}{1-\beta}}$ for any value of δ. Again, we will use this equation to find the right value for $\tilde{N}$ once we have chosen ε. When choosing such ε, we will have to respect the constraint $0 \leq \frac{\varepsilon}{1-\beta} \leq 1$ in order to make condition (d) hold. Hence any ε such that $0 < \varepsilon \leq 1 - \beta$ satisfies condition (d), and $\beta < 1$ guarantees that such an ε exists.

As in the previous case, condition (a) is easily verified by $\alpha = 1$. For verifying condition (b) we choose $\varepsilon, \varepsilon', \tilde{N}, \tilde{M}$ such that conditions (ã), (b̃) and (d) are verified. Then we choose $N = \lceil \tilde{N} \rceil = 1$ and $M = \lceil \tilde{M} \rceil = \lceil n^{\frac{\varepsilon}{1-\beta}} \rceil$ so that condition (c) is verified. The analysis of condition (b) is analogous to the previous case and yields $N^{\delta-1} M^{\beta-1} n^2 \leq \left(n^{\frac{\varepsilon}{1-\beta}} + 1 \right)^{\beta-1} \cdot n^2 = O(n^{2-\varepsilon})$, which verifies condition (b).

Case 1.2: $\delta < 1$ and $\beta = 1$. In this case condition (b̃) simplifies to

$$\tilde{N}^{\delta-1} n^2 = n^{2-\varepsilon} \Leftrightarrow \tilde{N} = n^{\frac{\varepsilon}{1-\delta}},$$

where $1 - \delta > 0$ holds thanks to $\delta < 1$. Condition (ã) and condition (b̃) both concern $\tilde{N}$, and by combining them we obtain:

$$n^{\frac{\varepsilon}{1-\delta}} = n^{\frac{1-\varepsilon'}{\alpha-1}} \Leftrightarrow \frac{\varepsilon}{1-\delta} = \frac{1-\varepsilon'}{\alpha-1} \Leftrightarrow \varepsilon = \frac{1-\varepsilon'}{\alpha-1}(1-\delta).$$

We already know that $0 < \frac{1-\varepsilon'}{\alpha-1} \leq 1$, which guarantees that $0 < \varepsilon \leq 1 - \delta$ and also verifies condition (d). Indeed, condition (d) requires $0 \leq \frac{\varepsilon}{1-\delta} \leq 1$, but this is already kept in check by the fact that $\frac{\varepsilon}{1-\delta} = \frac{1-\varepsilon'}{\alpha-1}$. Since $\delta < 1$, we have $1 - \delta > 0$, and hence we can conclude that all conditions (ã), (b̃) and (d) hold.

Using Fact 1 we can prove that when choosing N as in (c) condition (a) holds.

$$\begin{aligned} N^{\alpha-1} n &= \lceil n^{\frac{1-\varepsilon'}{\alpha-1}} \rceil^{\alpha-1} n \\ &= O(n^{\frac{1-\varepsilon'}{\alpha-1}\alpha-1} n) \\ &= O(n^{2-\varepsilon'}). \end{aligned}$$

Observing that $\beta = 1$ makes condition (b) simplify to $N^{\delta-1} n^2 = O(n^{2-\varepsilon})$, and we can perform a similar analysis to obtain $N^{\delta-1} n^2 \leq \left(n^{\frac{\varepsilon}{1-\delta}} + 1 \right)^{\delta-1} \cdot n^2 = O(n^{2-\varepsilon})$, which verifies condition (b).

Case 1.3: $\delta = 1$ and $\beta < 1$. Similarly to the previous case, from condition (b̃) we get:

$$\tilde{M}^{\beta-1} n^2 = n^{2-\varepsilon} \Leftrightarrow \tilde{M} = n^{\frac{\varepsilon}{1-\beta}}.$$

Here, condition (d) is equivalent to $0 \leq \frac{\varepsilon}{1-\beta} \leq 1$, which is guaranteed by choosing ε such that $0 < \varepsilon \leq 1 - \beta$. Thus, conditions (ã), (b̃) and (d) hold. Assuming condition (c) we can perform a similar analysis to the previous case and conclude that conditions (a) and (b) also hold.