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, Helsinki, Finland
{massimo.equi,veli.makinen,alexandru.tomescu}@helsinki.fi

Abstract. The string matching problem on a node-labeled graph $G = (V, E)$ asks whether a given pattern string P has an occurrence in G, in the form of a path whose concatenation of node labels equals P. This is a basic primitive in various problems in bioinformatics, graph databases, or networks, but only recently proven to have a $O(|E||P|)$-time lower bound, under the Orthogonal Vectors Hypothesis (OVH). We consider here its *indexed* version, in which we can index the graph in order to support time-efficient string queries.

We show that, under OVH, no polynomial-time indexing scheme of the graph can support querying P in time $O(|P| + |E|^{\delta}|P|^{\beta})$, with either $\delta < 1$ or $\beta < 1$. As a side-contribution, we introduce the notion of *linear independent-components (lic) reduction* , allowing for a simple proof of our result. As another illustration that hardness of indexing follows as a corollary of a *lic* 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 indexed version, improving the recent result of Cohen-Addad, Feuilloley and Starikovskaya (SODA 2019), but with a slightly different boundary condition.

Keywords: Exact pattern matching · Indexing · Orthogonal vectors · Complexity theory · Reductions · Lower bounds · Edit distance · Graph query

1 Introduction

1.1 Background

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

Problem 1 (SMLG).

This work was partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 851093, SAFEBIO) and by the Academy of Finland (grants No. 309048, 322595, 328877).

T. Bureš et al. (Eds.): SOFSEM 2021, LNCS 12607, pp. 608–622, 2021.
https://doi.org/10.1007/978-3-030-67731-2_44

INPUT: A directed graph $G = (V, E, \ell)$, where $\ell : V \to \Sigma$ is a function assigning to every node $v \in V$ a label $\ell(v)$ over an alphabet Σ, and a pattern string $P \in \Sigma^+$.
OUTPUT: *True* if and only if there is a path[1] $(v_1, v_2, \ldots, v_{|P|})$ 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 (see references in Equi et al. [19]). 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 *pangenome*) that encodes all genomes of a population [17,25,34,40].

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 $O(|V| + |E|\lceil |P|/w \rceil)$ [39] 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 [10] 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. [19] by showing that the problem remains quadratic under OVH even for directed acyclic graphs (DAGs) that are *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 *indexed* 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. This setting is motivated by the fact that typically the large input graph is static (e.g. a pangenome graph), while new query patterns are produced continuously (e.g. genome fragments from sequencing).

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 linear time to support sub-quadratic time queries. Even before the OVH-based reductions, another weak lower bound was known to hold conditioned on the *Set Intersection Conjecture (SIC)* [12,29,38] (see also Table 1). Recent results [28] also underlined the link between OV and SIC, and presented an index for

[1] Notice that if we further require that the path repeats no node (i.e. is a *simple* path) then SMLG becomes NP-hard, since the Hamiltonian path problem can be easily reduced to it, see e.g. [19].

Table 1. Upper bounds (first four rows) and conditional lower bounds for indexed SMLG on a graph $G = (V, E)$ and a pattern P. In this table, W is the maximum length of a node label, $f(\cdot)$ is an arbitrary function, N is the total length of an elastic degenerate string and n is the number of degenerate symbols. Recall that it is NP-complete to decide whether a given graph is a Wheeler graph, while for all the other graph types recognition is not harder than indexing.

Graph	Indexing time	Query time	Reference, Year
Path	$O(\|E\|)$	$O(\|P\|)$	Classical [18]
Tree	$O(\|E\|)$	$O(\|P\|)$	[23], 2009
Wheeler graph	$O(\|E\|)$	$O(\|P\|)$	[24,42], 2014
Segment repeat-free founder block graph	$O(W\|E\|)$	$O(\|P\|)$	[35], 2020
DAG	$O(\|E\|^\alpha)$, $\alpha < 2$	$f(\|P\|)$ impossible under SIC	[12], 2013
Arbitrary graph	$O(\|E\|)$	$O(\|P\| + \|E\|^\delta \|P\|^\beta)$, $\delta + \beta < 2$ impossible under OVH	[10], 2016
Deterministic DAG	$O(\|E\|)$	$O(\|P\| + \|E\|^\delta \|P\|^\beta)$, $\delta + \beta < 2$ impossible under OVH	[19], 2019
Elastic degenerate string	$O(N^\alpha)$	$O(n^\delta \|P\|^\beta)$, $\delta < 1$ or $\beta < 1$ impossible under OVH	[26], 2020
Deterministic DAG	$O(\|E\|^\alpha)$	$O(\|P\| + \|E\|^\delta \|P\|^\beta)$, $\delta + \beta < 2$ impossible under OVH	Theorem 2
Arbitrary graph	$O(\|E\|^\alpha)$	$O(\|P\| + \|E\|^\delta \|P\|^\beta)$, $\delta < 1$ or $\beta < 1$ impossible under OVH	Theorem 3

OV which is polynomial in the number of the vectors, but exponential in their length.

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 $O(|P| + |E|^{1/2}|P|^2)$. 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 [42] proposed an extension of the *Burrows-Wheeler transform* [15] for prefix-sorted graphs. Standard indexing techniques [22,30,37] can be applied on such generalized Burrows-Wheeler transform to support linear-time pattern search, but the size of the transform can be exponential in the worst case. There have been some advances in making the approach more practical [25,34,41], but the exponential bottleneck has remained.

The concept of prefix-sorted graphs was later formalized into a more general concept of *Wheeler graphs* [24]: Conceptually these are a class of graphs

that admit a generalization of the Burrows-Wheeler transform, and thus a linear size index 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] gave polynomial time solutions on some special cases and improved the prefix-sorting algorithm to work in near-optimal time in the size of the output. Another special type of DAGs, called *Degenerate String*, [6,31] and its *Elastic* variant [32], have recently received more attention due to improvements on the online approaches [8,11] for performing string matching. They might be thought of being a promising candidate for indexing thanks to their relatively constrained structure. Nevertheless, this type of DAGs has been proved to be hard to index [26], as we discuss below in more detail.

Mäkinen et al. [35] recently showed that another special graph class, *segment repeat-free founder block graphs*, not directly characterized by the Wheeler property, can be indexed in linear time for linear-time queries. Without these special conditions on graphs, the indexing problem is still widely open.

In this paper we refute the existence of a polynomial indexing scheme for graphs able to provide sub-quadratic [before it said "linear"] time queries, under OVH. Our result holds even for deterministic DAGs with labels from a binary alphabet.

We should note that the results of this paper, via the preprint version of this work [20], have already been used by Gibney [26] while the current paper was under review. In particular, Gibney proved that no index built in polynomial time can provide subquadratic-time queries for elastic degenerate strings, by using our Theorem 5 as [26, Lemma 1]. Even if an elastic degenerate string can be represented as a labeled DAG, our direct approach implies a stronger hardness result on DAGs. Namely, hardness holds also for DAGs labeled using a *binary* alphabet, and in which the total degree[2] is at most 3.

Table 1 and Fig. 1 summarize the complexity landscape of indexed SMLG.

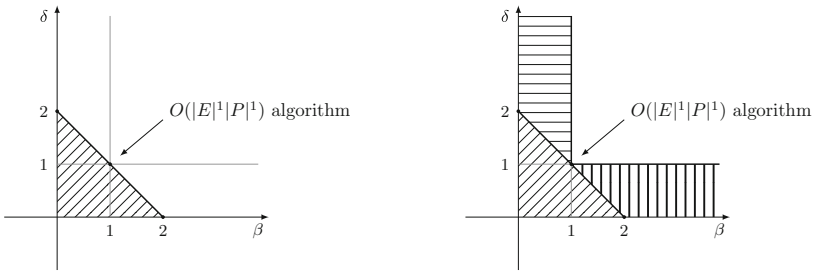
1.2 Results

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 $\epsilon > 0$, no algorithm can solve OV in time $O(N^{2-\epsilon}\text{poly}(d))$.

Notice that OVH [43] is implied by the better known *Strong Exponential Time Hypothesis* (SETH) [33], which states that for any constant $\epsilon > 0$, no algorithm can solve CNF-SAT in time $O(2^{(1-\epsilon)n})$, where n is the number of Boolean variables. Hence, 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. [2,4]. Recall the reduction from k-SAT to OV from [43]: the n variables of the formula ϕ are split into two groups of $n/2$

[2] The total degree is the sum of in-degree and out-degree of a node.



(a) $\alpha = 1$ (b) $\alpha \in \mathbb{R}$

Fig. 1. The dashed areas of the plots represent the forbidden values of δ and β for $O(|P| + |E|^{\delta}|P|^{\beta})$-time queries, under OVH. Figure 1a shows the lower bound that follows from the online case [10,19], and holds for $\alpha = 1$. Figure 1b depicts our lower bound (tight, thanks to the online $O(|E||P|)$-time algorithm from [7]) from Theorem 3. In addition, these hold for any value of α.

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 in polynomial time to support $O(M^{2-\epsilon}\text{poly}(d))$-time queries for any set Y of M vectors, for some $\epsilon > 0$. One now can adjust the splitting of the variables based on the hypothetical ϵ: the first part (corresponding to X) has $n\delta_\epsilon$ variables, and the other part (corresponding to Y) has $n(1 - \delta_\epsilon)$ 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 $\gamma > 0$, contradicting SETH.

In this paper, instead of employing this technique inside the reduction for indexed 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). Thus, 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 bounds for DTWD [1,14] and Frechet distance [13] are *lic* reductions, hence these problems automatically obtain a lower bound also for their indexed version.

On the other hand, the reductions for edit distance [9] and longest common subsequence [1,14] would need to be slightly tweaked to make the definition of *lic* reduction apply. Nevertheless, the features that are preventing these reductions from being *lic* concern only the size of the gadgets used and not their structural properties. Hence, we are confident that the modifications needed to such gadgets require only a marginal effort.

Moreover, the common indexed variation of the edit distance asks 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 [9] from OV to edit distance, this problem is utilized as an intermidiate step, and up to this point their reduction is indeed a *lic* reduction (Sect. 2.2). Hence, we immediately obtain the following result.

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

For $\delta = 1$ and $\beta = 1$ this lower bound is tight because there exists a matching online algorithm [36]. Theorem 1 also complements the recent result of Cohen-Addad, Feuilloley and Starikovskaya [16], 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 Sect. 3 we revisit the reduction from [19] and observe that it is a *lic* reduction. As such, we can immediately obtain the following result.

Theorem 2. *For any $\alpha, \beta, \delta > 0$ 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 $O(|P| + |E|^{\delta}|P|^{\beta})$, 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.*[3]

For $\delta = 1$ and $\beta = 1$ this lower bound is tight because there exists a matching online algorithm [7]. However, this bound does not disprove a hypothetical polynomial indexing algorithm with query time $O(|P|+|E|^{\delta}|P|^{2})$, 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.

[3] We implicitly assumed here that the graph G is the part of the input to be indexed. 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 $\alpha, \beta, \delta > 0$, with either $\beta < 1$ or $\delta < 1$, 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 $O(|P| + |E|^{\delta}|P|^{\beta})$, unless* OVH *is false.*

We obtain Theorem 3 by slightly modifying the reduction of [19] with the introduction of certain cycles, that allow for query patterns of length longer than the graph size. The theorem statement could be made slightly stronger by retaining some of the constrain on the graph that we had in Theorem 2, but we leave these details for the extended version of this work. See Sect. 3 for a brief discussion. We leave as open question whether the lower bound from Theorem 3 holds also for DAGs. See Sect. 3 for technical details on the difficulties of this special case.

Open Problem 1 *Does there exist $\alpha, \beta, \delta > 0$, with $\beta < 1$ or $\delta < 1$, 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 $O(|P| + |E|^{\delta}|P|^{\beta})$?*

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 $Q(|p_x|, |p_y|) = O(k^{O(1)}(|p_y| + |p_x|^{\delta}|p_y|^{\beta}))$. If this holds only when $k = O(1)$, 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 $A \leq^k_{lic} B$, 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 $\alpha, \beta, \delta > 0$, if $A \leq^k_{lic} B$ 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 $O(k^{O(1)}|a_y| + |b_x|^\delta |b_y|^\beta) = O(k^{O(1)}|a_y| + k^{O(1)}|a_x|^\delta |a_y|^\beta)$. 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. □

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 Sect. 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 found, the indexing lower bound follows directly.

Corollary 1. *Any problem P such that $\mathsf{OV} \leq^d_{lic} P$ holds is not (α, δ, β)-polynomially indexable, for any $\alpha, \beta, \delta > 0$, 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, where $ed(S_1, S_2)$ denotes the edit distance between string S_1 and S_2.

Problem 2 (PATTERN).

INPUT: Two strings T and P.
OUTPUT: $\min_{S \text{ substring of } T} ed(S, P)$.

Backurs and Indyk [9] 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, 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:

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" of the vectors, 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. Moreover, we care to take into account also the case $\alpha \leq 1$. In this way we rule out special cases like, for instance, $\delta < \alpha \leq 1$, which would leave the door open for efficient algorithms when $|E| \gg |P|$.

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 size is n. We partition set X into subsets of N vectors each, and set Y into subsets of M vectors each. Each pair of vector sets (X_i, Y_j) in which X_i is such a subset of X and Y_j is such a subset of Y constitutes an instance of (N, M)-OV. Solving all

the (X_i, Y_j) instances clearly solves the original problem.[4] Given a pair (X_i, Y_j), since we are assuming that (N, M)-OV is (α, δ, β)-polynomially indexable with parameter d, we can build an index on X_i in $O(d^{O(1)}(dN^{\alpha}))$ time and answer a query for Y_j in $O(d^{O(1)}(dN)^{\delta}(dM)^{\beta})$ time. Hence, by building a new index for every X_i and querying every Y_j we can cover all the pairs. Since we build $\lceil \frac{n}{N} \rceil$ indexes and we perform $\lceil \frac{n}{N} \rceil \lceil \frac{n}{M} \rceil$ queries, one for each pair (X_i, Y_j), the total cost for solving the original OV problem is:

$$O\left(d^{O(1)}\left((dN)^{\alpha}\frac{n}{N} + (dN)^{\delta}(dM)^{\beta}\frac{n}{N}\frac{n}{M}\right)\right) \tag{1}$$

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

In order to achieve a contradiction with OVH, we need such time complexity to be subquadratic in the original OV instance. Namely, it should be $O(d^{O(1)}(n^{2-\epsilon'} + n^{2-\epsilon}))$, for some $\epsilon, \epsilon' > 0$. Clearly, it must also hold $1 \leq N \leq n$ and $1 \leq M \leq n$, and N and M should be integers. 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 $\epsilon', \epsilon > 0, N$ and M such that:

$$\begin{array}{ll} \text{(a) } N^{\alpha-1}n = O(n^{2-\epsilon'}) & \text{(b) } N^{\delta-1}M^{\beta-1}n^2 = O(n^{2-\epsilon}) \\ \text{(c) } \quad N \in \mathbb{N},\ M \in \mathbb{N} & \text{(d) } \ 1 \leq N \leq n,\ 1 \leq M \leq n \end{array}$$

The solutions to this system differ depending on the values of α, δ and β. In Table 2 we present an exhaustive list of solutions for any choice of these parameters. The complete analysis on how to find the range of possible solutions to the system is presented in preprint version of this work [21].

Table 2. The solutions to the system for any given value of α, δ and β.

α	β	δ	N	M	ϵ'	ϵ
$\alpha < 2$	$\beta < 1$	Any δ	1	n	1	$1-\beta$
	$\beta \geq 1$	$\delta < 1$	n	1	$2-\alpha$	$1-\delta$
$\alpha \geq 2$	$\beta < 1$	Any δ	1	n	1	$1-\beta$
	$\beta \geq 1$	$\delta < 1$	$\lceil n^{\frac{1}{2(\alpha-1)}} \rceil$	1	$\frac{1}{2}$	$\frac{1-\delta}{2(\alpha-1)}$

We conclude that depending on α, δ and β we find ourselves into one of the listed cases and thus we can always find valid values for ϵ, ϵ', N and M that lead to an algorithm for OV running in time $O(n^{2-\epsilon} + n^{2-\epsilon'})$, contradicting OVH. □

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

[4] 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.

3 Indexing Labeled Graphs for String Matching

We are now left to prove Theorem 2 and Theorem 3. While the former can be obtained solely applying the concept of *lic* reduction, the latter requires more attention.

Recall the following conditional lower bound for SMLG from Equi et al. [19].

Theorem 6 ([19]). *For any $\epsilon > 0$, SMLG on labeled deterministic DAGs cannot be solved in either $O(|E|^{1-\epsilon}\,|P|)$ or $O(|E|\,|P|^{1-\epsilon})$ time unless OVH fails. 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.*

In order to prove Theorem 2 it is enough to check the structure of the reduction used to prove Theorem 6 in [19].

Proof (Theorem 2). Given an OV instance with sets X and Y, the reduction from [19] 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. [5] This shows that the two conditions of the linear independent-components reduction property hold, thus OV $\leq^{d}_{lic}$ SMLG. We conclude the proof by directly applying Corollary 1. □

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 [19] 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.

Proof (Theorem 3). Given an (N, M)-OV instance with sets X and Y, we first show how it is possible to modify the reduction form [19] to fit an arbitrarily long pattern into the graph. The desired lower bound will then follow by applying the *lic* reduction.

The pattern that we use is the same pattern P of the original reduction, which is built over 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 $Y = \{y_1, \ldots, y_M\}$. Namely, we define $P = \mathtt{bb}P_{y_1}\mathtt{e}\,\mathtt{b}P_{y_2}\mathtt{e}\ldots\mathtt{b}P_{y_M}\mathtt{ee}$, 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 $h \in \{1, \ldots, d\}$, such that $P_{y_i}[h] = \mathtt{1}$ if and only if $y_i[h] = 1$.

The graph G' that we need to make our new reduction work is depicted in Fig. 2 and we now discuss how it can be built. 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 $x_j \in X$. Each component $G_W^{(j)}$ can be constructed so that the following holds.

[5] Originally [19] P and G were built on X and Y, respectively. Since it is immaterial for correctness, we assumed the opposite here to keep in line with the notation.

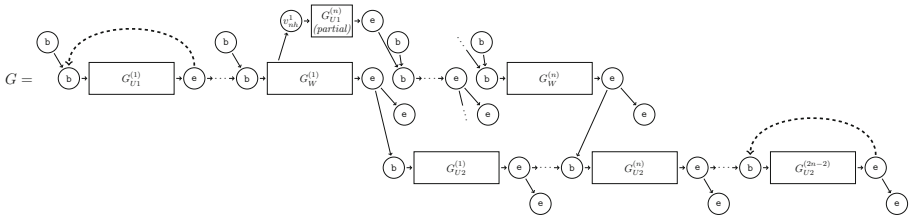


Fig. 2. Non-deterministic graph G'. We add the dashed thick edges, absent in the acyclic graph from [19], to handle (N, M)-OV instances with $M > N$.

Lemma 2 ([19]). *Subpattern* $\texttt{b}P_{y_i}\texttt{e}$ *has a match in* G_W *if and only if there exist* $x_j \in X$ *such that* $x_i \cdot y_j = 0$.

In addition, we need a universal gadget $G_U = (V_U, E_U, L_U)$ of $2N - 2$ components $G_U^{(k)}$. We build such components in the same way as in [19], and for the correctness of the current proof it suffices to know that each component can match any subpattern P_{y_i}. Let us now build the same final graph G as in [19] by using two instances G_{U1} and G_{U2} of G_U and merging the first one of these with G_W. The resulting graph G has total size $O(dN)$ and corresponds to the one shown in Fig. 2 without the dashed edges. Note that every path from a pair of consecutive b-nodes to a pair of consecutive e-nodes passes through a $G_W^{(j)}$ component. Indeed, this graph satisfies the following property.

Lemma 3 ([19]). *Pattern* P *has a match in* G *if and only if a subpattern* $\texttt{b}P_{x_i}\texttt{e}$ *of* P *has a match in the underlying subgraph* G_W *of* G_{U1W}.

From graph G we then build final graph G' with the addition of 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 $G_{U2}^{(2N-2)}$. This final graph G' is the one of Fig. 2.

The intuition on how the reduction works 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 bb prefix and an ee suffix. Our new edges allow to accommodate (N, M)-OV instances with $M > N$.

Formally, we need to prove that Lemma 3 holds also if considering G' and the case $N \neq M$.

Proof (Lemma 3 for G', $N \neq M$). For the ($\Rightarrow$) implication, we follow the same logic as in [19] and we observe that pattern P needs to start a match only by using a pair of consecutive b-nodes and such a match can be completed only by using a pair of consecutive e-nodes after having matched a $G_W^{(j)}$ component. Hence we can use Lemma 2 to ensure that there exists a pair of orthogonal vectors. The ($\Leftarrow$) implication is easier: if a subpattern $\texttt{b}P_{x_i}\texttt{e}$ of P has a match in the underlying subgraph G_W then we can match the prefix of P preceding

$\mathtt{b}P_{x_i}\mathtt{e}$ in G_U since every subpattern $\mathtt{b}P_{x_{i'}}\mathtt{e}$, $1 \leq i' < i$, can be matched in the $G_{U1}^{(i)}$ components. Observe that if $N < M$ then there might not be enough such components to match all the subpatterns of $|P|$. In that case, our newly added backward edges can be used to match the component $G_{U1}^{(1)}$ multiple times. The $N > M$ case poses no problem. The same reasoning applies to the subpatterns $\mathtt{b}P_{x_{i''}}\mathtt{e}$, $i < i'' \leq M$, constituting the suffix of $|P|$. Such subpatterns can be matched in G_{U2} possibly exploiting our backward edge. □

Since Lemma 3 still holds, we conclude that the reduction using graph G' works for any value of N and M. Hence, applying Corollary 2, we obtain Theorem 3. □

As a final note, we informally describe how some of the features of Theorem 2 could be kept also in Theorem 3. We leave a more detailed and formal discussion for the extended version of this work.

Remark 1. The statement of Theorem 3 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. Hence, with respect to Theorem 2, we have to drop only determinism. Indeed, the two additional edges that we added to build our final graph G' respect the degree constraint. Moreover, applying the same transformation as in [19], the theorem holds even when we are restricted to use a binary alphabet.

References

1. Abboud, A., Backurs, A., Williams, V.V.: Tight hardness results for LCS and other sequence similarity measures. In: FOCS 2015, Berkeley, CA, USA, pp. 59–78 (2015)
2. Abboud, A., Rubinstein, A., Williams, R.R.: Distributed PCP theorems for hardness of approximation in P. In: IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), Berkeley, CA, USA, pp. 25–36. IEEE (2017)
3. Abboud, A., Williams, R., Yu, H.: More applications of the polynomial method to algorithm design. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, San Diego, California, pp. 218–230 (2015)
4. Abboud, A., Williams, V.V.: Popular conjectures imply strong lower bounds for dynamic problems. In: IEEE 55th Annual Symposium on Foundations of Computer Science, Philadelphia, PA, USA, pp. 434–443 (2014)
5. Alanko, J., D'Agostino, G., Policriti, A., Prezza, N.: Regular languages meet prefix sorting. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Salt Lake City, UT, USA, pp. 911–930 (2020)
6. Alzamel, M., et al.: Degenerate string comparison and applications. In: Parida, L., Ukkonen, E. (eds.) 18th International Workshop on Algorithms in Bioinformatics (WABI 2018). Leibniz International Proceedings in Informatics (LIPIcs), vol. 113, pp. 21:1–21:14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018)
7. Amir, A., Lewenstein, M., Lewenstein, N.: Pattern matching in hypertext. In: Dehne, F., Rau-Chaplin, A., Sack, J.-R., Tamassia, R. (eds.) WADS 1997. LNCS, vol. 1272, pp. 160–173. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63307-3_56

8. Aoyama, K., et al.: Faster online elastic degenerate string matching. In: Annual Symposium on Combinatorial Pattern Matching (CPM 2018), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2018)
9. Backurs, A., Indyk, P.: 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, New York, USA, pp. 51–58 (2015)
10. Backurs, A., Indyk, P.: Which regular expression patterns are hard to match? In: IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), New Brunswick, NJ, USA, pp. 457–466. IEEE (2016)
11. Bernardini, G., Gawrychowski, P., Pisanti, N., Pissis, S.P., Rosone, G.: Even faster elastic-degenerate string matching via fast matrix multiplication. In: Baier, C., Chatzigiannakis, I., Flocchini, P., Leonardi, S. (eds.) 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9–12, 2019, Patras, Greece. LIPIcs, vol. 132, pp. 21:1–21:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2019)
12. Bille, P.: Personal Communication at Dagstuhl Seminar on Indexes and Computation over Compressed Structured Data (2013)
13. Bringmann, K.: Why walking the dog takes time: frechet distance has no strongly subquadratic algorithms unless seth fails. In: IEEE 55th Annual Symposium on Foundations of Computer Science, pp. 661–670. IEEE (2014)
14. Bringmann, K., Kunnemann, M.: Quadratic conditional lower bounds for string problems and dynamic time warping. In: IEEE 56th Annual Symposium on Foundations of Computer Science, Washington, USA, pp. 79–97. IEEE (2015)
15. Burrows, M., Wheeler, D.: A block sorting lossless data compression algorithm. Tech. Rep. 124, Digital Equipment Corporation (1994)
16. Cohen-Addad, V., Feuilloley, L., Starikovskaya, T.: Lower bounds for text indexing with mismatches and differences. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, San Diego, USA, pp. 1146–1164 (2019)
17. Consortium, T.C.P.G.: Computational pan-genomics: status, promises and challenges. Briefings in Bioinform. **19**(1), 118–135 (2018)
18. Crochemore, M., Rytter, W.: Jewels of Stringology. World Scientific (2002)
19. Equi, M., Grossi, R., Mäkinen, V., Tomescu, A.I.: On the complexity of string matching for graphs. In: 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019), Patras, Greece, pp. 55:1–55:15 (2019)
20. Equi, M., Grossi, R., Tomescu, A.I., Mäkinen, V.: On the complexity of exact pattern matching in graphs: determinism and zig-zag matching. arXiv e-prints arXiv:1902.03560 (2019)
21. Equi, M., Mäkinen, V., Tomescu, A.I.: Graphs cannot be indexed in polynomial time for sub-quadratic time string matching, unless seth fails. arXiv e-prints arXiv:2002.00629 (2020)
22. Ferragina, P., Manzini, G.: Indexing compressed texts. J. ACM **52**(4), 552–581 (2005)
23. Ferragina, P., Luccio, F., Manzini, G., Muthukrishnan, S.: Compressing and indexing labeled trees, with applications. J. ACM **57**(1), 4:1–4:33 (2009)
24. Gagie, T., Manzini, G., Sirén, J.: Wheeler graphs: a framework for BWT-based data structures. Theor. Comput. Sci. **698**, 67–78 (2017)
25. Garrison, E., et al.: Variation graph toolkit improves read mapping by representing genetic variation in the reference. Nat. Biotechnol. **36**, 875 (2018)

26. Gibney, D.: An efficient elastic-degenerate text index? not likely. In: Boucher, C., Thankachan, S.V. (eds.) SPIRE 2020. LNCS, vol. 12303, pp. 76–88. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59212-7_6
27. Gibney, D., Thankachan, S.V.: On the hardness and inapproximability of recognizing Wheeler graphs. In: ESA 2019, Munich/Garching, Germany, pp. 51:1–51:16 (2019)
28. Goldstein, I., Lewenstein, M., Porat, E.: Orthogonal vectors indexing. In: ISAAC 2017, Dagstuhl, Germany, pp. 40:1–40:12 (2017)
29. Goldstein, I., Lewenstein, M., Porat, E.: On the hardness of set disjointness and set intersection with bounded universe. In: ISAAC 2019, Shanghai, China. LIPIcs, vol. 149, pp. 7:1–7:22 (2019)
30. Grossi, R., Vitter, J.: Compressed suffix arrays and suffix trees with applications to text indexing and string matching. SIAM J. Comput. **35**(2), 378–407 (2006)
31. Grossi, R., et al.: On-line pattern matching on similar texts. In: CPM 2017. vol. 78, p. 1. Schloss Dagstuhl-Leibniz-Zentrum für Informatik GmbH (2017)
32. Iliopoulos, C.S., Kundu, R., Pissis, S.P.: Efficient pattern matching in elastic-degenerate texts. In: Drewes, F., Martín-Vide, C., Truthe, B. (eds.) LATA 2017. LNCS, vol. 10168, pp. 131–142. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-53733-7_9
33. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. J. Comput. Syst. Sci. **62**(2), 367–375 (2001)
34. Kim, D., Paggi, J.M., Park, C., Bennett, C., Salzberg, S.L.: Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. Nat. Biotechnol. **37**(8), 907–915 (2019)
35. Mäkinen, V., Cazaux, B., Equi, M., Norri, T., Tomescu, A.I.: Linear time construction of indexable founder block graphs. In: WABI 2020, Pisa, Italy. LIPIcs, vol. 172, pp. 7:1–7:18 (2020). https://doi.org/10.4230/LIPIcs.WABI.2020.7
36. Masek, W.J., Paterson, M.S.: A faster algorithm computing string edit distances. J. Comput. Syst. Sci. **20**(1), 18–31 (1980)
37. Navarro, G., Mäkinen, V.: Compressed full-text indexes. ACM Comput. Surv. **39**(1), 2 (2007)
38. Patrascu, M., Roditty, L.: Distance oracles beyond the Thorup-Zwick bound. SIAM J. Comput. **43**(1), 300–311 (2014)
39. Rautiainen, M., Mäkinen, V., Marschall, T.: Bit-parallel sequence-to-graph alignment. Bioinformatics **35**(19), 3599–3607 (2019)
40. Schneeberger, K., et al.: Simultaneous alignment of short reads against multiple genomes. Genome Biol. **10**, R98 (2009)
41. Sirén, J.: Indexing variation graphs. In: ALENEX 2017, Barcelona, Spain, pp. 13–27 (2017)
42. Sirén, J., Välimäki, N., Mäkinen, V.: Indexing graphs for path queries with applications in genome research. IEEE/ACM Trans. Comput. Biol. Bioinform. **11**(2), 375–388 (2014)
43. Williams, R.: A new algorithm for optimal 2-constraint satisfaction and its implications. Theor. Comput. Sci. **348**(2–3), 357–365 (2005)