Conditional Lower Bounds for String Matching in Labelled Graphs

Massimo Equi
Aalto University, Finland

Abstract

The problem of *String Matching in Labelled Graphs (SMLG)* is one possible generalization of the classic problem of finding a string inside another of greater length. In its most general form, SMLG asks to find a match for a string into a graph, which can be directed or undirected. As for string matching, many different variations are possible. For example, the match could be exact or approximate, and the match could lie on a path or a walk. Some of these variations easily fall into the NP-hard realm, while other variants are solvable in polynomial time. For the latter ones, fine-grained complexity has been a game changer in proving quadratic conditional lower bounds, allowing to finally close the gap with those upper bounds that remained unmatched for almost two decades.

If the match is allowed to be approximate, SMLG enjoys the same conditional quadratic lower bounds shown for example for edit distance (Backurs and Indyk, STOC '15). The case that really requires ad hoc conditional lower bounds is the one of finding an *exact* match that lies on a *walk*. In this work, we focus on explaining various conditional lower bounds for this version of SMLG, with the goal of giving an overall perspective that could help understand which aspects of the problem make it quadratic. We will introduce the reader to the field of fine-grained complexity and show how it can successfully provide the exact type of lower bounds needed for polynomial problems such as SMLG.

2012 ACM Subject Classification Theory of computation → Pattern matching; Theory of computation → Problems, reductions and completeness

Keywords and phrases conditional lower bounds, strong exponential time hypothesis, fine-grained complexity, string matching, graphs

Digital Object Identifier 10.4230/OASIcs.Grossi.2025.7

Category Research

Funding This work was supported in part by the Research Council of Finland, Grant 359104.

1 Introduction

The classic problem of string matching consists in finding a match for a shorter pattern string P into a longer text string T. This problem has been extensively studied throughout the years, but the core fundamental complexity result was already discovered in the '70s, when the problem was proven to be solvable in linear time [19]. The problem of *String Matching in Labelled Graphs* (SMLG) is a generalization of the string matching problem, where we look for matches of P not in a text but in a graph with nodes labelled by single characters. The SMLG problem first received attention in the '90s [20, 3, 4] due to potential applications in network searches and, later on, applications such as bioinformatics [15], graph databases [5] and heterogeneous networks [24] motivated it anew. Finding quadratic upper bounds of the form $O(|E||P|)$ was possible for different variations of the problem [4, 21, 25], and those results were later proven to be optimal under some complexity hypotheses for other problems [10, 11]. This conditional lower bounds falls into the field of the so-called *fine-grained complexity*, and in this work we aim to showcase how fine-grained complexity lower bounds are achieved and why they changed the game for polynomial problems, in particular for SMLG.

The journey of fine-grained complexity begins in the early 2000s, when Impagliazzo, Paturi and Zane [17, 18] put forward a hypothesis on the complexity of SAT in conjunctive normal form (CNF-SAT). The hypothesis was named *Strong Exponential Time Hypothesis* (SETH), and states that no algorithm can solve a CNF-SAT instance over n variables in time $O(2^{\alpha n})$, where $\alpha < 1$. This hypothesis still holds its ground, since no one has been able to disprove it as of the writing of this work. A few years later, Williams [26] showed how SETH can be used to prove conditional lower bounds for polynomial problems. Among others, this work presented a fine-grained reduction from CNF-SAT to the *Orthogonal Vectors* (OV) problem, that is a reduction that requires subexponential time, which in this case was $O(2^{\frac{n}{2}})$[1]. In the OV problem, we are given sets X and Y of n binary vectors of dimension d, and we are asked to find a pair (x, y) of orthogonal vectors $x \in X$ and $y \in Y$. So far, no algorithm was able to solve OV in $O(n^{2-\varepsilon}\text{poly}(d))$ time, and this fact is referred to as the *Orthogonal Vectors Conjecture* (OVC). Thus, the reduction between CNF-SAT and OV shows that a $O(n^{2-\varepsilon}\text{poly}(d))$ time algorithm for OV would disprove SETH, or SETH $\Rightarrow$ OVC, and this is an important connection between exponential and polynomial complexities. Indeed, this allows to prove SETH-based lower bounds by reducing from OV instead of CNF-SAT, a choice that usually makes reductions cleaner and the lower bounds more reliable as, in principle, OVC might be true even if SETH fails.

After this initial connection, many other lower bounds for polynomial problems conditioned on SETH have been shown. Among these, one of the most celebrated examples is probably the lower bound for the problem of computing *edit distance* (EDIT) between two strings A and B [6]. The textbook algorithm for EDIT solves the problem in $O(n^2)$ time, where $n = |A| = |B|$, and has been known for decades, but the first lower bound was achieved only thanks to a fine-grained reduction from OV. This brings us back to SMLG, as the conditional lower bound for EDIT has consequences also for matching strings in graphs. Indeed, there are different variations of SMLG, and one thing to specify is whether the matches for pattern P in graph G should be exact or approximate, and whether walks are admitted (same nodes can be matched multiple times) or only paths (every node must be matched ones). To allow for approximate matches, one has to specify what an approximate match is. We could be very permissive, and allowing edit operations to occur both in the pattern and in the graph. This possibility was explored and proven to be an NP-hard problem [4]. If we then ask for a path spelling a string at minimum edit distance from P, then we incur in the following problem: if we label every node with character a and we query for pattern of length $|P| = |V|$, then we are solving the Hamiltonian Path problem, known to be NP-hard. Thus, let us look for a *walk* spelling a path at minimum edit distance from P. In this setting, an $O(|E||P|)$ algorithm exists [21, 25], and since a string can be viewed as a chain of nodes, EDIT is a special case of this problem, and thus the conditional quadratic lower bound matches the upper bound.

As previously mentioned, the version of SMLG where we ask for a walk in G spelling an exact match for P has also received a conditional quadratic lower bound [10, 11]. This result however could not be achieved as a simple adaptation of other lower bounds, and needed a

[1] We acknowledge that this use of the term “subexponential” is improper, as a real subexponential complexity should be of the form $O(2^{o(n)})$. Nonetheless, we will use this wording for lack of a better term.

custom fine-grained reduction from OV. Moreover, this highlights also a remarkable difference between the text and graph case: while the problem is linear for texts, it is quadratic for graphs, while in the approximate case it is quadratic for both texts and graphs.

At this point, one natural approach to find upper bounds could be to change the setting to try to play around the lower bound. For example, one could ask if queries to the pattern could be solved efficiently, that is in subquadratic time, after spending quadratic or even polynomial time indexing the graph. Unfortunately, also this question was answered negatively, as Equi, Mäkinen and Tomescu [11] proved that superpolynomial indexing time is needed in order to achieve subquaratic time queries. This result was given as a special case of a more general technique, which allows to claim that the same type of indexing lower bound holds for any problem with a fine-grained reduction from OV respecting a certain structure.

All these fine-grained results where further strengthened after the introduction of new hardness hypotheses [1] believed to be more reliable than SETH. Using this framework, Gibney, Hoppenworth, and Thankachan [16] showed that the hardness of SMLG can be based on these hypotheses, and this improved the lower bound proving that even shaving logarithmic factors from the quadratic time complexity is hard. This enhanced lower bound is given by reducing from a problem called Formula-SAT, a generalisation of CNF-SAT, and due to the nature of the problem the design of the reduction had to be modified into a structure of recursively-defined gadgets.

As a final note, although this work is centred around the quadratic lower bounds for SMLG, it is important to remember that there are graphs for which the problem is easier to solve, and even linear time complexity can be achieved. This is the case for instance for Wheeler graphs [14], certain (Elastic) Founder graphs [23, 13], trees [22] and Funnels [7]. More in general, it is also possible to parametrize the complexity in a parameter expressing the "sortability" of the graph [9, 8].

In the sections that follow, we will introduce SMLG more formally as well as the fine-grained complexity hypotheses. As a warm up, we show how to reduce CNF-SAT to OV in subexponential time, as the reduction is very central to the field and very concise at the same time. Then, we give an overview of the main ideas behind different fine-grained reductions for SMLG that were given throughout the years, with the goal of showcasing the different features among them.

2 Problem Definition

A pattern string P is a string of characters drawn from an alphabet Σ, has length $|P|$, and $P[i]$ is its i-th character. We say that $G = (V, E, \ell)$ is a labeled graph if V is a set of nodes, E a set of edges over V, and $\ell : V \to \Sigma$ is a labeling function that associate a character of alphabet Σ to every node in V. Moreover, we say that P has a *match* in G if there is a walk of nodes $v_1, \ldots, v_m$ in G such that $P = \ell(v_1) \cdots \ell(v_m)$. Then, SMLG is formally defined as follows.

▶ **Problem 1** (String Matching in Labelled Graphs (SMLG)).
INPUT: *A labeled graph $G = (V, E, \ell)$ and a pattern string P, both over alphabet Σ.*
OUTPUT: True *if and only if P has at least one match in G.*

Note that in this definition of SMLG we ask for a *walk* to exist and not a *simple path*. This is because, if we are not allowed to repeat nodes, there is a straightforward reduction from the Hamiltonian path problem, making the problem NP-Complete. To see this, simply define $\Sigma = \{\mathtt{A}\}$, $P = \mathtt{A}\ \mathtt{A} \cdots \mathtt{A}$, $|P| = |V|$ and $\ell(v) = \mathtt{A}$ for every $v \in V$. In other words, we are asking to find a match for a path as long as the number of nodes, and if we can match every node only once, we are finding an Hamiltonian path.

2.1 Some Upper Bounds for SMLG

When faced with an algorithmic problem, one of the first questions we want to answer is often how bad does the brute force algorithm performs. If by brute force we just mean trying all possibilities, that is check all paths of length $|P|$, then we get very poor results, as in general there are $O(|V|^{|P|})$ many such paths. However, better alternatives exit, so let us briefly give a bird's-eye view of these approaches, without going into details. A quadratic algorithm for solving SMLG in the exact setting in general graphs has been known since the '90s [3, 4]. This algorithm constructs the product graph obtained between the graph and the pattern, which is obtained by repeating every node in the graph as many times as there are characters in the patterns, and then placing edges according to the original structure of the graph plus the constraint of having node labels appear in the pattern at specific positions. Then, the algorithm performs a DFS visit of the graph which, if successful in reaching a certain target node, reveals a match of P in G. Not long after, a first quadratic solution [21], which was later refined [25], was found also for the approximate setting, when we are looking for a walk spelling a string at minimum edit distance from P. Moreover, there are linear time algorithms for a rooted (thus directed) tree [2], and even when the roots of many trees are connected in a cycle [11]. This highlights already some important topology features: SMLG is linear in trees, while in DAGs it is quadratic in general. Non-topological characterization can also be given. For instance, in Wheeler graphs [14] a node can be sorted according to the sets of strings that it represents, that is the set of all possible strings spelled by paths reaching that node. This implies a total order on the nodes, which leads to linear time matching algorithms. This idea can be generalized and the time complexity can be parametrized as a function of the sortability of the graph [8].

3 Fine-Grained Complexity

Quadratic algorithms have been the best we could achieve for general formulations of SMLG, and the reason is because better solutions are unlikely to exist. In order to show that we have reached the optimum, we of course need lower bound techniques, and here is where fine-grained complexity comes into play, has it provides us the means of proving such lower bounds via reductions. Thus, before presenting the conditional lower bounds for SMLG, we first introduce fine-grained complexity tools and basic notions needed to understand the later reductions. The central hypothesis in fine-grained complexity is the *Strong Exponential Time Hypothesis* (SETH).

▶ **Definition 2** (SETH). *For every $\varepsilon > 0$, there exists $k \geq 3$ such that no deterministic or randomized algorithm can solve an instance of CNF-SAT over n variables with clauses of size at most k in $O(2^{(1-\varepsilon)n})$ time.*

This hypothesis is called *strong* to remark that it is a stronger statement when compared to the more forgiving *Exponential Time Hypothesis* (ETH), which forbids the existence of algorithms solving CNF-SAT in $O(2^{o(n)})$ time. To give an example, if an algorithm solves CNF-SAT in $O(2^{\frac{n}{2}})$, SETH is violated but ETH is not.

There are polynomial problems for which hardness conjectures are independently believed, and for which there also exist efficient reductions from CNF-SAT. When possible, it is preferred finding a reduction from these problems instead of reducing directly from CNF-SAT. This way, multiple hypotheses have to fail to invalidate the conditional lower bound, and moreover we have to deal only with polynomial complexities in the analysis, instead of having both exponential and polynomial complexities.

$$F = \overset{c_1}{(v_1 \lor \neg v_3)} \quad \land \quad \overset{c_2}{(\neg v_1 \lor v_2)} \quad \land \quad \overset{c_3}{(\neg v_1 \lor v_3 \lor \neg v_4)} \quad \land \quad \overset{c_4}{(v_2 \lor v_4)}$$

$$\begin{array}{ccccccc} & & v_1\, v_2 & c_1\, c_2\, c_3\, c_4 & & & \\ & a_1 = & (0\ \ 0) & (1\ \ 0\ \ 0\ \ 1) & = x_1 & & x_i[h] = 0 \\ 2^{\frac{n}{2}} \Bigg\{ & a_2 = & (0\ \ 1) & (1\ \ 0\ \ 0\ \ 0) & = x_2 & \Bigg\} \ m & \Leftrightarrow \\ & a_3 = & (1\ \ 0) & (0\ \ 1\ \ 1\ \ 1) & = x_3 & & a_i \models c_h \\ & a_4 = & (1\ \ 1) & (0\ \ 0\ \ 1\ \ 0) & = x_4 & & \\ & & v_3\, v_4 & c_1\, c_2\, c_3\, c_4 & & & \\ & b_1 = & (0\ \ 0) & (0\ \ 1\ \ 0\ \ 1) & = y_1 & & y_j[h] = 0 \\ 2^{\frac{n}{2}} \Bigg\{ & b_2 = & (0\ \ 1) & (0\ \ 1\ \ 1\ \ 0) & = y_2 & \Bigg\} \ m & \Leftrightarrow \\ & b_3 = & (1\ \ 0) & (1\ \ 1\ \ 0\ \ 1) & = y_3 & & b_j \models c_h \\ & b_4 = & (1\ \ 1) & (1\ \ 1\ \ 0\ \ 0) & = y_4 & & \\ & & \underbrace{\hphantom{(0\ \ 0)}}_{\frac{n}{2}} & \underbrace{\hphantom{(0\ \ 0\ \ 0\ \ 0)}}_{d} & & & \end{array}$$

Figure 1 An example of the reduction from CNF-SAT to OV. Formula F has $n = 4$ variables and $d = 4$ clauses, and we construct two sets of $m = 2^{\frac{n}{2}} = 4$ vectors of size $d = 4$.

The polynomial problems most commonly used as a base for reductions to other polynomial problems are Orthogonal Vectors, All Pairs Shortest Paths, and 3-SUM [27]. In this work, we focus solely on *Orthogonal Vectors* (OV) which, intuitively, asks the following: given two sets of binary vectors, answer whether it is possible to find a vector in the first set and a vector in the second set so that they are orthogonal.

▶ **Definition 3** (OV). *Let $X, Y \subseteq \{0,1\}^d$ be two sets of $n = |X| = |Y|$ binary vectors each of length $d = \omega(\log n)$. Determine whether there exist $x \in X$ and $y \in Y$ such that $x \cdot y = \sum_{i=1}^{d} x[i] \cdot y[i] = 0$.*

The notation $x[i] \cdot y[i]$ indicates the scalar product when used for two single entries of vectors x and y, while it refers to the dot product $x \cdot y$ when applied on the vectors themselves. Currently, it is conjectured that no algorithm can solve OV in truly subquadratic time.

▶ **Definition 4** (OVC). *For every constant $\varepsilon > 0$, no deterministic or randomized algorithm can solve OV over two sets of n binary vectors of size d in $O(n^{2-\varepsilon} poly(d))$ time.*

One central result in fine-grained complexity is that SETH and OVC are connected via a subexponential reduction. We give a proof of this result, as we find it easy to follow and very instructive at the same time.

▶ **Lemma 5.** *SETH implies OVC.*

Proof. We can prove the contrapositive by showing a reduction from CNF-SAT to OV, an example of which is given in Figure 1. In other words, we prove that a subquadratic-time algorithm for OV implies a subexponential-time algorithm for CNF-SAT. Consider an instance of CNF-SAT where formula F has d clauses over n variables $v_1, \ldots, v_n$. The idea is to generate two sets of vectors that can represent partial truth assignments of F using only $O(2^{\frac{n}{2}})$ space, imposing the property that finding a pair of orthogonal vectors reveals how to combine two partial truth assignments into an actual truth assignment satisfying F.

To this end, we evenly split the variables into two sets $V_1 = \{v_1, \ldots, v_{\frac{n}{2}}\}$ and $V_2 = \{v_{\frac{n}{2}+1}, \ldots, v_n\}$. Then, we define sets A and B as the set of every possible partial truth assignment for the variables in V_1 and V_2, respectively. Given that $|V_1| = |V_2| = \frac{n}{2}$, we have that $|A| = |B| = 2^{\frac{n}{2}}$. Now we can define our two sets of vectors X and Y, consisting of $m = \frac{n}{2}$ vectors each. For every assignment $a_i \in A$, we construct a vector in X as follows. We evaluate the variables in V_1 under a_i, and if this is enough to satisfy clause c_h (i.e. at least one literal evaluates to 1), then the h-th entry of vector x_i shall be $x_i[h] = 0$, otherwise $x_i[h] = 1$. We perform the same construction for Y using B and V_2.

The logic of the reduction is the following. Our goal is to find a truth assignment satisfying F combining two partial truth assignments $a_i \in A$ and $b_j \in B$ encoded as vectors $x_i \in X$ and $y_j \in Y$. Since the h-th entry of vector x_i corresponds to the h-th clause of F, if the entry is $x_i[h] = 0$ then the clause is satisfied by a_i, meaning that whatever value $y_j[h]$ is, it does not change the end result. Conversely, if $x_i[h] = 1$, then a_i does not satisfy the h-th clause, and we need $y_j[h] = 0$ to guarantee that b_j satisfies it instead.

Now observe that the reduction takes $O(2^{\frac{n}{2}}\mathrm{poly}(d))$ time, as there are $2 \cdot 2^{\frac{n}{2}}$ of size d, where d is the number of clauses. At this point, it is easy to see that, if vectors x_i and y_j are orthogonal, and thus for every h either $x_i[h] = 0$ or $y_j[h] = 0$, then at least one of the two partial truth assignments $a_i \in A$ or $b_j \in B$ satisfies clause c_h, and vice versa. Thus, F is satisfied if and only if there exist $x \in X$ and $y \in Y$ such that $x \cdot y = 0$. To conclude the proof, assume OVC is false, namely there exists an algorithm that solves OV in $O(m^{2-\alpha}\mathrm{poly}(d))$ time, for some $\alpha > 0$. Then, this would provide an algorithm for CNF-SAT running in time

$$O(2^{\frac{n}{2}(2-\alpha)}\mathrm{poly}(d)) = O(2^{n(1-\frac{\alpha}{2})}\mathrm{poly}(d)),$$

Taking $\varepsilon = \frac{\alpha}{2}$ proves SETH false. ◀

The connection between SETH and OVC shown by Lemma 5 allow us to obtain lower bounds conditioned on both by only reducing from OV. This is advantageous, because conditional lower bounds obtained in this way stop being valid only when both SETH and OVC fail. Moreover, for polynomial problems, reducing from OV makes proofs cleaner as we can reason only in terms of polynomial time complexities, without having to juggle them together with exponential-time complexities.

4 The Lower Bound for String Matching in Labelled Graphs

Using fine-grained reductions, we can provide a quadratic lower bound for *exact* SMLG conditioned on SETH and OVC. In this section, we state this lower bound in its simplest form, focusing on giving the intuition behind the structure of the reduction. In Section 5, we explain a few different ways of strengthening this result.

The conditional lower bound for *exact* SMLG is formally stated as follows.

▶ **Theorem 6.** *The String Matching in Labelled Graphs (SMLG) problem on pattern string P and graph $G = (V, E)$ cannot be solved neither in $O(|E|^{1-\varepsilon}\,|P|)$ nor in $O(|E|\,|P|^{1-\varepsilon})$ time for any constant $\varepsilon > 0$, unless SETH fails.*

We now sketch the idea of the reduction from OV used to show the lower bound [12, 10]. Starting from sets of vectors X and Y, the reduction builds pattern P and graph G in $O(nd)$ time, such that P matches in G if and only if there is a pair of orthogonal vectors between X and Y. We first describe how to construct the pattern, and then the graph.

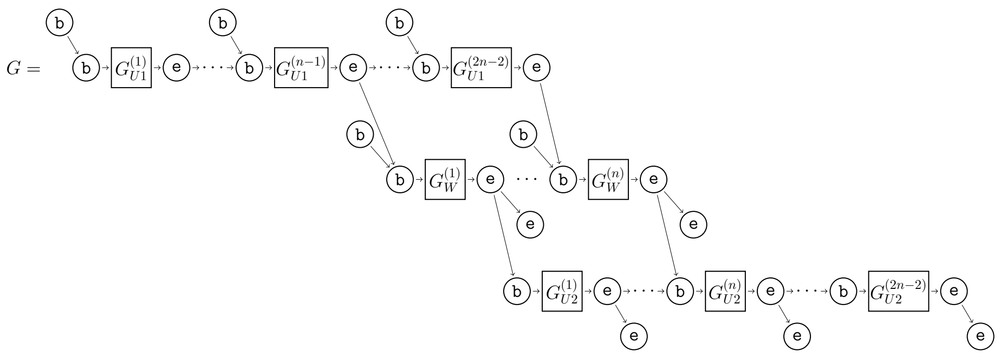

Figure 2 A high-level view of graph G constructed by the reduction from OV. The top and bottom rows consists of universal gadgets, while in the middle rows there are only vector gadgets, which match only patterns corresponding to orthogonal vectors. Here, we also add an orientation on the edges, even if it is not explicitly mentioned in the statement of Theorem 6. Indeed, this is allowed since the b- and e-nodes already force an orientation, as stated in Theorem 10. Figure adapted from [12].

Patter P is defined over alphabet $\Sigma = \{\texttt{b}, \texttt{e}, \texttt{0}, \texttt{1}\}$, has length $|P| = O(nd)$, and can be built in $O(nd)$ time from the first set of vectors $X = \{x_1, \ldots, x_n\}$. Namely, we define

$$P = \texttt{bb}P_{x_1}\texttt{e}\,\texttt{b}P_{x_2}\texttt{e}\ldots\texttt{b}P_{x_n}\texttt{ee}$$

where P_{x_i} is a string of length d that is just a copy of $x_i \in X$, that is $x_i[h] = 0 \Rightarrow P_{x_i}[h] = \texttt{0}$ and $x_i[h] = 1 \Rightarrow P_{x_i}[h] = \texttt{1}$, for $1 \leq i \leq n$ and $1 \leq h \leq d$. Here we are using characters b and e to mark the beginning and the end of each subpattern P_{x_i}, respectively. Substrings bb and ee instead respectively mark the beginning and the end of the entire pattern, forcing it to start and end the match at specific locations in the graph.

Graph G is built on top of the second set of vectors $Y = \{y_1, \ldots, y_n\}$, and consists of different substructures hierarchically organized: gadgets encoding single entries of the vectors are combined into gadgets encoding entire vectors, which are further combined to form the whole graph. At a macroscopic level, we want to structure the graph in three conceptual rows stacked on top of each other, as exemplified in Figure 2. In the graph, characters b and e force the subpatterns to correctly align to the gadgets encoding entire vectors, and thus forcing the entire match to synchronize at the subpattern level. Then, the idea is to make the top and bottom rows able to match any properly-synchronizing prefix and suffix of the pattern, respectively, while the middle row shall match only those subpatterns encoding vectors that are orthogonal to some vector in Y. Thus, if we force a match to start from the top or middle row, and end in the middle or bottom row, at least one subpattern must match in the middle row, which will be possible only if the pair of vectors encoded by that subpattern and the graph gadget it matches in are orthogonal. As shown in Figure 2, we can force this behaviour by introducing paths of two nodes spelling the string bb in the top and middle row. In pattern P, this string is present only as a prefix, marking the beginning of the pattern. The same logic applies to paths of two nodes spelling the string ee in the middle and bottom row, since that string is present in P only as a suffix.

As already mentioned, the single graph gadgets in the middle row G_W must provide the following property.

▶ **Lemma 7.** *Subpattern* $\texttt{b}P_{x_i}\texttt{e}$ *has a match in* $G_W^{(j)}$ *if and only if* $x_i \cdot y_j = 0$.

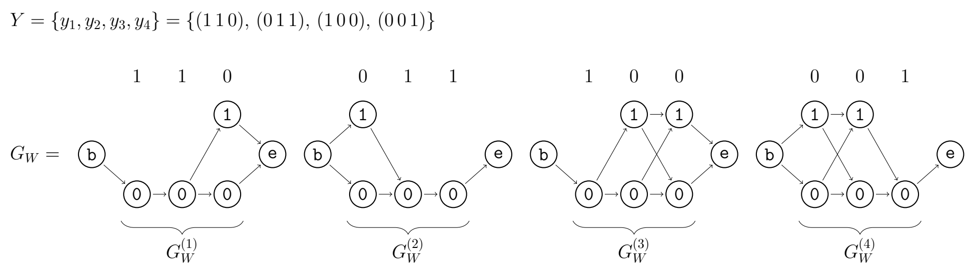
$Y = \{y_1, y_2, y_3, y_4\} = \{(1\,1\,0), (0\,1\,1), (1\,0\,0), (0\,0\,1)\}$
1 1 0
0 1 1
1 0 0
0 0 1
$G_W =$
b
e
0
1
$G_W^{(1)}$
$G_W^{(2)}$
$G_W^{(3)}$
$G_W^{(4)}$

Figure 3 Vector gadgets matching only patterns corresponding to orthogonal vectors. For instance, $G_W^{(3)}$ will match 010 or 011, but not 110, as $(110) \cdot (100) \neq 0$. Figure adapted from [12].

Let us see how to construct such gadgets. Since $\sum_{h=1}^{d} x[h] \cdot y[h] = 0$ when every term of the sum is 0, the idea is to make the graph force the pattern to have character $S[h] = 0$ when $y[h] = 1$, while letting $S[h]$ be either 0 or 1 when $y[h] = 0$. To achieve this, first we place a path of h nodes all with label 0, then we add a node with label 1 for those positions such that $y[h] = 0$, and finally we fully connect nodes encoding the entry at position h to those for position $h+1$, for $1 \leq h \leq d$. Figure 3 shows an example of the entire middle row G_W, where gadget $G_W^{(j)}$ encodes vector y_j, for $1 \leq j \leq n$, using nodes with labels b and e to mark the beginning and the end of the gadgets. It then holds the following.

▶ **Lemma 8.** *Subpattern* bP_{x_i}e *has a match in G_W if and only if there exists $y_j \in Y$ such that $x_i \cdot y_j = 0$.*

In order to complete the reduction, it remains only to build the universal gadgets for the top and bottom rows of G. This is easily achieved by following the same construction scheme as for gadgets $G_W^{(j)}$, and just placing both a 0-node and a 1-node at every position. Finally, we remark that in order to make all possible "shifts" of the pattern possible, the top and bottom rows of G must have $2(n-1)$ universal gadgets. We can now claim the following.

▶ **Lemma 9.** *Pattern P has a match in G if and only if a subpattern* bP_{x_i}e *of P has a match in subgraph G_W.*

Notice that every gadget $G_W^{(j)}$ or universal gadget consists of $O(d)$ nodes and edges, and there are $O(2(n-1) + n + 2(n-1)) = O(n)$ such gadgets, for a total size of $O(nd)$. This consideration together with Lemma 7, 8 and 9 let us claim Theorem 6.

5 Tighter Lower Bounds

After having established a first quadratic conditional lower bound for SMLG, we now want to see how far we can push it. What if there are constraints on the graph, or on the alphabet? Can we make up for the quadratic complexity if we first build an index? Can we condition the lower bound on other hypotheses to make it even stronger? Let us explore these questions.

5.1 DAGs, Determinism and Bounded Degree

In this section, we study how strong assumptions we can make on the graph while still having the conditional quadratic lower bound hold. Starting from the graph structure, we remark that the nodes labelled with b and e already force the pattern to follow a specific direction, that is the pattern never needs to visit a node twice to make the reduction work. Indeed, directing the edges from left to right and from top to bottom as in Figure 4 does not compromise the construction.

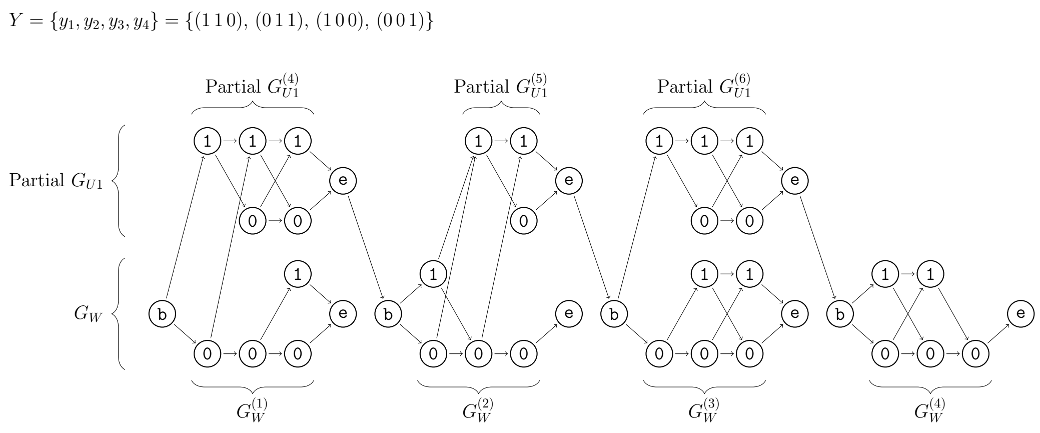

Figure 4 New gadget G_{U1W} obtained by merging G_W with part of gadget G_{U1}. The pattern can "escape" to the G_{U1} part to match a prefix until it finds a suitable $G_W^{(j)}$ where to match a subpattern encoding an orthogonal vector. Figure adapted from [12].

Nevertheless, there is one major non-trivial limitation to the reduction scheme proposed in Section 4. Even after making the edges directed, there are nodes that have two out-neighbours with the same label. For instance, nodes labelled with `e` in the top row might have two out-neighbours labelled with `e`. This might suggest that forcing every out-neighbour to have different labels could make the graph close enough to a DFA so that the problem becomes easier. Perhaps surprisingly, there is a way to modify the reduction so that it covers even this case. The construction is shown in Figure 4. Intuitively, we merge the top and middle row of G together, with the idea being that the pattern tries to match every sub-pattern in the underlying G_W as much as possible. When the match cannot be continued, the pattern will momentarily escape to a partial universal gadget for the remaining of the current sub-pattern. The non-determinism is removed by the fact that the nodes labelled `b` in the top and middle row are now merged together.

One last concern is the degree of the graph. By looking at the graph that we obtain after merging the top and middle row, we notice that every node is never connected to more than three other nodes. In a DAG, we can capture this concept by saying that the sum of outdegree and indegree of any node is at most three.

We are now ready to state the conditional lower bound for SMLG in its strongest form.

▶ **Theorem 10.** *The String Matching in Labeled Graphs (SMLG) problem on pattern string P and labelled* deterministic *directed acyclic graph (DAG) $G = (V, E)$ cannot be solved neither in $O(|E|^{1-\varepsilon}\,|P|)$ nor in $O(|E|\,|P|^{1-\varepsilon})$ time unless SETH fails, for any constant $\varepsilon > 0$. This holds even if it is restricted to a binary alphabet and to graphs in which the sum of outdegree and indegree of any node is at most three.*

In this statement, we are claiming that the theorem holds also for a binary alphabet. Indeed, there exists an binary encoding that allows to cover also this case. This requires to take care of some technicalities that we do not find fitting for this work, and hence we refer the reader to the literature [12].

5.2 Indexing Lower Bound

Given the results of previous sections, there seems to be no hope of solving SMLG in less than quadratic time, as long as SETH holds. But rules are made to be broken, or at least circumvented. In the lower bounds shown so far, we always analyzed the setting in which we have to solve the problem from ground zero, without building any data structure to help us in performing queries. Hence, one question arises: can we break through the quadratic complexity if we build some kind of index on the graph? In other words, can we pay a quadratic (or even greater) cost upfront during preprocessing, so that the complexity reduces at query time? Although tantalizing, this option is also ruled out [11].

▶ **Theorem 11.** *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 OVC 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.*

For $\delta = 1$ and $\beta = 1$ this lower bound is tight because there exists a matching online algorithm [3, 4]. 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 in practical applications graphs are much larger than the pattern, such an algorithm would be quite significant for small enough δ. However, when the graph is allowed to have cycles, we also show that this is impossible under OVC.

▶ **Theorem 12.** *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 OVC is false.*

Theorem 12 is obtained by slightly modifying the reduction of [10] with the introduction of certain cycles that allow querying patterns of length longer than the graph size. As for the online SMLG lower bound, it can be proven that this results holds also when restricted to a binary alphabet, and for graphs in which the sum of out-degree and in-degree of any node is at most three.

Without diving into technical details, we remark that this result is given as an application of a more general technique involving *linear independent-components* reductions [11], combined with some improvements on folklore knowledge about OV indexability. Intuitively, a linear independent-components reduction is a reduction from OV performed so that the instance of the output problem can be separated in two parts, each one depending only on one of the two sets of vectors. For instance, in the case of SMLG, we observe that that the reduction builds the pattern using only the first set of vectors, and the graph using only the second.

This independence property of the two components is crucial when combined with the following fact about OV. Suppose that in a OV instance we partition sets X and Y of n vectors into many subsets $X_1, \ldots, X_{\frac{n}{N}}$ and $Y_1, \ldots, Y_{\frac{n}{N}}$ where $N = |X_i| = |Y_j|$, so that each (X_i, Y_j) is a smaller OV instance and, moreover, solving all such instances also solves OV for the original X and Y. If we claim that we can index all the X_i in polynomial time to answer queries to the Y_j in subquadratic time, then we contradict OVC. This is because it is always possible to choose a small enough N such that both the total indexing time and the total query time fall below quadratic. For example, assuming we can index X_i in time $O(N^{\alpha})$, we can take $N = O(n^{\frac{1}{\alpha}})$. Then, given that we have $\frac{n}{N}$ instances (X_i, Y_j), the total indexing time is $O(\frac{n}{N}N^{\alpha}) = O(n^{2-\frac{1}{\alpha}})$, and the total query time is bound to be subquadratic in n if every Y_j can be queried in time subquadratic in N.

Using the properties of linear independent-components reductions, we can use indexes for SMLG to answer queries for OV, and thus we can transfer the indexing lower bounds from OV to SMLG as in Theorem 11 and 12. Moreover, the consequences of this reduction scheme can be further strengthened if one considers a generalized version of OV where X and Y can have different cardinalities [11].

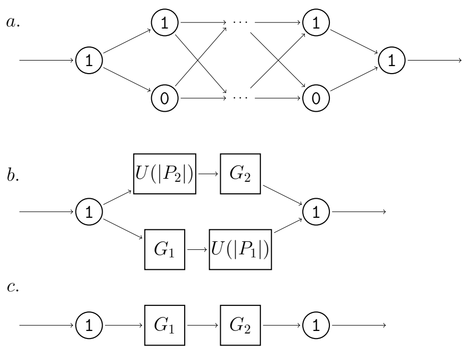

Figure 5 The universal gadget of variable length (a), the gadget encoding gate $\vee$ (b) and the gadget encoding gate $\wedge$ (c). Subgraphs G_1 and G_2 represent two gadgets encoding two subtree of the formula, respectively, that are the left and right children of an $\vee$ (b) gate or an $\wedge$ (c) gate. Figure adapted from Gibney et al. [16].

5.3 Lower Bounds from Formula-SAT for shaving Logarithmic Factors

All lower bounds shown for SMLG are conditional, and the strength of a conditional lower bound comes from the reliability of its hypothesis. It is then desirable to always look for more believable hypotheses on which we can base our conditional lower bounds. This is what was done in the work by Abboud et al. [1], where the starting problems of the proposed reductions are more general versions of the SAT problem, rather than CNF-SAT. This allows not only to base their conditional lower bounds on more reliable hypotheses, but also to achieve better complexity bounds. Later, Gibney et al.[16] showed how to use such techniques to obtain better lower bounds for problems like SMLG.

▶ **Theorem 13.** *If SMLG can be solved in time $O\left(\frac{|E||P|}{\log^c |E|}\right)$ or $O\left(\frac{|E||P|}{\log^c |P|}\right)$ for all $c > 0$, then NTIME$[2^{O(n)}]$ does not have non-uniform polynomial-size log-depth circuits.*

This result is achieved by showing a subexponential-time reduction from the Formula-SAT problem to SMLG. To define the Formula-SAT problem, let us first define a deMorgan formula. A deMorgan formula is a Boolean formula that can be represented as a binary tree where the leaves represents a variable or its negation, while internal nodes represent either one of the logical operators $\{\wedge, \vee\}$. Notice that negation is allowed only at the leaf level. The Formula-SAT problem is then the satisfiability problem over a deMorgan formula.

We now show the key ideas of the reduction that differ from the one based on OV. First, here we reduce from a SAT problem over n variables and thus, as for showing SETH $\Rightarrow$ OVC, we consider two sets of partial truth assignments, each one defined on $\frac{n}{2}$ variables. Next, since the formula is structured as a tree, the operators $\wedge$ and $\vee$ can be nested, and hence the reduction should have a recursive structure. The base case of the recursion are the gadgets that encode the input gates. Here, some care is needed to avoid having to deal with negation, but we defer the details to the work of Gibney et al. [16], where they use an intermediate problem to solve this issue. For building the recursive gadgets, an universal gadget of variable length $U(\mu)$ is needed. This has exactly the same structure of the universal gadgets in the reduction from OV, but with the two rows of 0- and 1-nodes elongated to accommodate matches for subpatterns of length μ. Then, the gadgets encoding gates $g = (g_1 * g_2)$, $* \in \{\wedge, \vee\}$, can be built by combining previously constructed gadgets for gates g_1 and g_2 as in Figure 5.

References

1 Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388. ACM, 2016. doi:10.1145/2897518.2897653.

2 Tatsuya Akutsu. A linear time pattern matching algorithm between a string and a tree. In Alberto Apostolico, Maxime Crochemore, Zvi Galil, and Udi Manber, editors, *Combinatorial Pattern Matching, 4th Annual Symposium, CPM 93, Padova, Italy, June 2-4, 1993, Proceedings*, volume 684 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1993. doi:10.1007/BFb0029792.

3 Amihood Amir, Moshe Lewenstein, and Noa Lewenstein. Pattern matching in hypertext. In Frank K. H. A. Dehne, Andrew Rau-Chaplin, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 5th International Workshop, WADS '97, Halifax, Nova Scotia, Canada, August 6-8, 1997, Proceedings*, volume 1272 of *Lecture Notes in Computer Science*, pages 160–173. Springer, 1997. doi:10.1007/3-540-63307-3_56.

4 Amihood Amir, Moshe Lewenstein, and Noa Lewenstein. Pattern matching in hypertext. *J. Algorithms*, 35(1):82–99, 2000. doi:10.1006/jagm.1999.1063.

5 Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39, February 2008. doi:10.1145/1322432.1322433.

6 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.*, 47(3):1087–1097, 2018. doi:10.1137/15M1053128.

7 Manuel Cáceres. Parameterized algorithms for string matching to dags: Funnels and beyond. In Laurent Bulteau and Zsuzsanna Lipták, editors, *34th Annual Symposium on Combinatorial Pattern Matching, CPM 2023, June 26-28, 2023, Marne-la-Vallée, France*, volume 259 of *LIPIcs*, pages 7:1–7:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.CPM.2023.7.

8 Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, and Nicola Prezza. Co-lexicographically ordering automata and regular languages - part I. *J. ACM*, 70(4):27:1–27:73, 2023. doi:10.1145/3607471.

9 Nicola Cotumaccio and Nicola Prezza. On indexing and compressing finite automata. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2585–2599. SIAM, 2021. doi:10.1137/1.9781611976465.153.

10 Massimo Equi, Roberto Grossi, Veli Mäkinen, and Alexandru I. Tomescu. On the complexity of string matching for graphs. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132 of *LIPIcs*, pages 55:1–55:15, 2019. doi:10.4230/LIPICS.ICALP.2019.55.

11 Massimo Equi, Veli Mäkinen, and Alexandru I. Tomescu. Graphs cannot be indexed in polynomial time for sub-quadratic time string matching, unless SETH fails. *Theor. Comput. Sci.*, 975:114128, 2023. doi:10.1016/J.TCS.2023.114128.

12 Massimo Equi, Veli Mäkinen, Alexandru I. Tomescu, and Roberto Grossi. On the complexity of string matching for graphs. *ACM Trans. Algorithms*, 19(3):21:1–21:25, 2023. doi:10.1145/3588334.

13 Massimo Equi, Tuukka Norri, Jarno Alanko, Bastien Cazaux, Alexandru I. Tomescu, and Veli Mäkinen. Algorithms and complexity on indexing elastic founder graphs. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021, Fukuoka, Japan*, volume 212 of *LIPIcs*, pages 20:1–20:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ISAAC.2021.20.

14 Travis Gagie, Giovanni Manzini, and Jouni Sirén. Wheeler graphs: A framework for bwt-based data structures. *Theor. Comput. Sci.*, 698:67–78, 2017. doi:10.1016/J.TCS.2017.06.016.

15 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, August 2018. doi:10.1038/nbt.422710.1038/nbt.4227.

16 Daniel Gibney, Gary Hoppenworth, and Sharma V. Thankachan. Simple reductions from formula-sat to pattern matching on labeled graphs and subtree isomorphism. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 232–242. SIAM, 2021. doi:10.1137/1.9781611976496.26.

17 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

18 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/JCSS.2001.1774.

19 Donald E. Knuth, James H. Morris Jr., and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977. doi:10.1137/0206024.

20 Udi Manber and Sun Wu. Approximate string matching with arbitrary costs for text and hypertext. In *Advances In Structural And Syntactic Pattern Recognition, Bern, Switzerland, 26-28 August 1992*, pages 22–33. World Scientific, 1992. doi:10.1142/9789812797919_0002.

21 Gonzalo Navarro. Improved approximate pattern matching on hypertext. *Theor. Comput. Sci.*, 237(1-2):455–463, 2000. doi:10.1016/S0304-3975(99)00333-3.

22 Kunsoo Park and Dong Kyue Kim. String matching in hypertext. In Zvi Galil and Esko Ukkonen, editors, *Combinatorial Pattern Matching, 6th Annual Symposium, CPM 95, Espoo, Finland, July 5-7, 1995, Proceedings*, volume 937 of *Lecture Notes in Computer Science*, pages 318–329. Springer, 1995. doi:10.1007/3-540-60044-2_51.

23 Nicola Rizzo, Massimo Equi, Tuukka Norri, and Veli Mäkinen. Elastic founder graphs improved and enhanced. *Theor. Comput. Sci.*, 982:114269, 2024. doi:10.1016/J.TCS.2023.114269.

24 Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. A survey of heterogeneous information network analysis. *IEEE Trans. Knowl. Data Eng.*, 29(1):17–37, 2017. doi:10.1109/TKDE.2016.2598561.

25 Mikko Rautiainen Tobias and Marschall. Aligning sequences to general graphs in $O(V + mE)$ time. *bioRxiv*, 2017. doi:10.1101/216127.

26 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/J.TCS.2005.09.023.

27 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pages 3447–3487, 2018. doi:10.1142/9789813272880_0188.