Quantum Time Complexity and Algorithms for Pattern Matching on Labeled Graphs

Parisa Darbari[1], Daniel Gibney[2], and Sharma V. Thankachan[3]

[1] Florida Polytechnic University, Lakeland, FL 33805, USA
pdarbarikozekanan@floridapoly.edu
[2] Georgia Institute of Technology, Atlanta, GA 30332, USA
daniel.j.gibney@gmail.com
[3] North Carolina State University, Raleigh, NC 27695, USA
svalliy@ncsu.edu

Abstract. The problem of matching (exactly or approximately) a pattern P to a walk in an edge labeled graph $G = (V, E)$, denoted PMLG, has received increased attention in recent years. Here we consider conditional lower bounds on the time complexity of quantum algorithms for PMLG as well as a new quantum algorithm. We first provide a conditional lower bound based on a reduction from the Longest Common Subsequence problem (LCS) and the recently proposed NC-QSETH. For PMLG under substitutions to the pattern, our results demonstrate (i) that a quantum algorithm running in time $O(|E|m^{1-\varepsilon} + |E|^{1-\varepsilon}m)$ for any constant $\varepsilon > 0$ would provide an algorithm for LCS on two strings X and Y running in time $\tilde{O}(|X||Y|^{1-\varepsilon} + |X|^{1-\varepsilon}|Y|)$, which is better than any known quantum algorithm for LCS, and (ii) that a quantum algorithm running in time $O(|E|m^{\frac{1}{2}-\varepsilon} + |E|^{\frac{1}{2}-\varepsilon}m)$ would violate NC-QSETH. Results (i) and (ii) hold even when restricted to binary alphabets for P and the edge labels in G. We then provide a quantum algorithm for all versions of PMLG (exact, only substitutions, and substitutions/insertions/deletions) that runs in time $\tilde{O}(\sqrt{|V||E|} \cdot m)$. This is an improvement over the classical $O(|E|m)$ time algorithm when the graph is non-sparse.

Keywords: Pattern Matching · Labeled Graphs · Quantum Algorithms.

1 Introduction

We consider an approximate version of the Pattern Matching on Labeled Graphs problem (PMLG) under substitutions to the pattern, defined as follows: Given a directed edge-labeled graph $G = (V, E)$ with alphabet Σ, a string P of length m also over alphabet Σ (which we call a pattern), and an integer $\delta \geq 0$, determine if there exists a walk in G that matches a string P' such that $d_H(P, P') \leq \delta$. Here, $d_H(P, P')$ denotes the Hamming distance between P and P' and a walk is a ordered list of edges in E, i.e., e_1, ..., e_m where e_i and e_{i+1} are incident to the same vertex for $1 \leq i < m$. Edges are allowed to be repeated in a walk.

Letting $label(e)$ denote the edge label for an edge $e \in E$, we say a length m string $P'[1, m]$ matches a walk e_1, ..., e_m if $P'[i] = label(e_i)$ for $1 \leq i \leq m$.

PMLG was first considered in the context of pattern matching in hypertext [5, 30, 32, 33]. It has become increasingly important in Computational Biology where labeled graphs are used as multi-genomic references and sets of reads obtained through sequencing must be mapped to the reference [1, 10, 13, 17, 27, 34]. PMLG is also used in variant calling [9, 12, 25] and read error correction [28, 31].

The theoretical aspects of PMLG have also received significant study. The classical algorithm for PMLG is a dynamic programming solution that runs in time $O(|E|m)$ [5, 32]. It was first shown in [15] that a PMLG algorithm running in time $O(|E|^{1-\varepsilon}m + |E|m^{1-\varepsilon})$ for any constant $\varepsilon > 0$ would contradict the Strong Exponential Time Hypothesis (SETH) even for directed acyclic graphs (DAGs) and $\delta = 0$. These results were later strengthened in [18] to show that the same lower bounds hold based on likely weaker assumptions in circuit complexity. Nevertheless, there exist classes of graphs where the exact matching problem can be solved in near-linear time, e.g., Wheeler graphs [16]. However, recognizing whether a given graph has these properties is a hard problem [4, 19, 20]. The version of the problem where modifications are allowed to labels in the graph rather than the pattern has also been considered, which is NP-hard even when restricted to only substitutions over binary alphabets on special classes of graphs [5, 21, 26].

Despite the extent of the applied and theoretical work, there has been sparse research on utilizing quantum computing to solve PMLG. Equi et al. [14] recently considered the problem for leveled DAGs, where they presented an algorithm running in $O(|E| + \sqrt{m})$. Several closely related problems have been studied as well. Aaronson et al. [2] look at quantum algorithms for the problem of determining whether a string is contained in a regular language were considered. However, these regular languages were represented as monoids rather than NFAs, meaning the input representation could differ drastically from the labeled graphs used here. For finding exact matches in a single string (which could be viewed as a path) there exists a quantum algorithm running in $\tilde{O}(\sqrt{n} + \sqrt{m})$ time[4] on a string of length n and pattern of length m [24, 36].

We provide the first hardness result for PMLG in the quantum computing setting based on a reduction from the Longest Common Subsequence problem (LCS) and the conjectured hardness NC-QSETH [8], along with a new algorithm yielding a quantum speedup for non-sparse graphs.

1.1 Quantum Computing and Input Model

Quantum algorithms typically have their problem instance expressed as an *oracle*, or a function that allows one to query the problem instance. On a quantum computer, these queries can be made with an input that is the superposition of multiple inputs, allowing for a type of parallelism. We refer the interested reader to [23]. These oracles are often treated as black boxes, but they can also

[4] $\tilde{O}(\cdot)$ suppresses poly-logarithmic factors.

be provided as a Boolean circuit, or through an algorithmic description (under the quantum random access assumption discussed more below). The query complexity of a quantum algorithm is defined as the number of times that the oracle gets queried by the algorithm, and the quantum time complexity is the number of elementary gates[5] needed to implement the quantum algorithm, in addition to the number of queries.

A lower-level description of a quantum algorithm in terms of unitary operators acting on a state vector (a quantum circuit) is necessary for many of the algorithms that are the fundamental building blocks of quantum computing. These include, for example, quantum random walk algorithms for finding marked vertices in a graph [29], period finding algorithms [35], and Grover's search [22]. However, it is often possible to utilize these fundamental algorithms on a higher level of abstraction. This accommodates algorithm descriptions more similar to those used in imperative programming. Examples of this approach include the graph algorithms presented in [11], the $\tilde{O}(\sqrt{n} + \sqrt{m})$ pattern matching algorithm mentioned in the introduction [24], and a recent algorithm for finding the longest common substring of two strings [3]. One useful assumption is quantum random access (described in [3, 7]). Using quantum random access, a classical T-time algorithm can be invoked by a quantum search algorithm, like Grover's search, in $O(T)$ time. We assume quantum random access here as well and provide our algorithm description at a high level. In fact, our solution in Section 3 can be seen as an algorithm (or even implemented as a small Boolean circuit) that utilizes the oracles of the original PMLG instance to create new oracles that are then used as input for a pre-existing quantum algorithm for shortest st-path in a directed graph.

For PMLG, we assume that our oracles allow us to query the indegree/outdegree and adjacency list of any vertex and the label of any edge. Any symbol in the pattern P can also be queried by specifying an index.

1.2 NC-QSETH

When establishing computational complexity results for quantum algorithms, using known lower bounds on query complexity has an immediate limitation for proving super-linear lower bounds on quantum time complexity. For problems where the input represents something such as a graph, once a linear number of queries have been made, the entire input is obtained by the algorithm. An alternative approach taken by the authors of [8] is to establish conditional lower bounds on quantum time complexity using the hypothesized hardness NC-QSETH. Although the details of NC-QSETH, which is based on a conjectured hardness of determining properties of circuits in a subset of the circuit class NC, are too complex to be covered here, we can use the result below.

Lemma 1 (LCS lower bounds based on NC-QSETH [8]). *Under NC-QSETH the Longest Common Subsequence problem (LCS) on two strings of length n cannot be solved in quantum time $\tilde{O}(n^{1.5-\varepsilon})$ for any constant $\varepsilon > 0$.*

[5] Elementary gates are defined in [6].

1.3 Our Results

We prove the following theorem in Section 2.

Theorem 1. *There exists a reduction from LCS with strings X and Y over alphabet Σ to PMLG with substitutions over a binary alphabet. This requires $O((|X|+|Y|)\log(|X|+|Y|)\cdot\log^2|\Sigma|)$ time (on a classical computer) and outputs a graph $G=(V,E)$ where $|V|,|E| = O(|X|\log(|X|+|Y|)\cdot\log^2|\Sigma|)$ and pattern $P[1,m]$ where $m = O(|Y|\log(|X|+|Y|)\cdot\log^2|\Sigma|)$.*

Theorem 1 gives us the following Corollaries.

Corollary 1. *An algorithm for PMLG with substitutions to the pattern over a binary alphabet running in quantum time $\tilde{O}(|E|^{1-\varepsilon}m+|E|m^{1-\varepsilon})$ for any constant $\varepsilon > 0$ would provide an algorithm running in quantum time $\tilde{O}(|X||Y|^{1-\varepsilon} + |X|^{1-\varepsilon}|Y|)$ for LCS.*

It should be noted that no strongly sub-quadratic quantum algorithms for LCS are known.

Corollary 2. *An algorithm for PMLG with substitutions to the pattern over a binary alphabet running in quantum time $\tilde{O}(|E|^{\frac{1}{2}-\varepsilon}m + |E|m^{\frac{1}{2}-\varepsilon})$ for any constant $\varepsilon > 0$, would provide an algorithm running in quantum time $\tilde{O}(|X|^{\frac{1}{2}-\varepsilon}|Y|+ |X||Y|^{\frac{1}{2}-\varepsilon})$ for LCS, violating NC-QSETH.*

In Section 3, we provide an algorithm running in quantum time $\tilde{O}(m\sqrt{|V||E|})$ for PMLG based on Durr et al.'s quantum algorithm for shortest path [11], implying a quantum speedup over the classical algorithm when the graph is not sparse, i.e., $|E| = \Omega(|V|^{1+\varepsilon})$. This algorithm also works when insertions and deletions are allowed to the pattern, in addition to substitutions.

2 Reduction from LCS to PMLG

We first present a simplified version of the reduction to PMLG with a larger alphabet and then show how to modify it to obtain the result for PMLG on binary alphabets. In the decision version of LCS, we are given two strings X, Y and $k \geq 0$ and have to decide where there exists a common subsequence of X and Y having a length at least k. Suppose $|Y| \geq |X|$ and let $n = |Y|$.

We construct our graph G based on the string X. We start by making two sets of vertices u_1, u_2, ..., $u_{|X|}$ and v_1, v_2, ..., $v_{|X|}$. We add directed edges (v_i, u_i) with labels $X[i]$ for $1 \leq i \leq |X|$. All remaining edges are labeled with a new symbol # that is not found in either X or Y. We then create edges (u_i, v_{i+1}) for $1 \leq i \leq |X| - 1$. Next, for v_i, $1 \leq i \leq |X|$ we create edges (v_i, v_i), (v_i, v_{i+1}), (v_i, v_{i+2}), (v_i, v_{i+4}), ..., (v_i, v_{i+2^c}) for the largest c such that $i + 2^c \leq |X|$ and the edge $(u_{|X|}, u_{|X|})$. See Figure 1. Let $\delta = n - k$ and

$$P = \#^{\lceil \log n \rceil + 1}\, Y[1]\, \#^{\lceil \log n \rceil + 1}\, Y[2]\, \#^{\lceil \log n \rceil + 1}\, \ldots\, \#^{\lceil \log n \rceil + 1}\, Y[n].$$

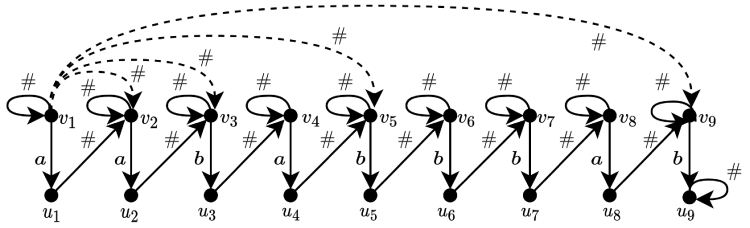

Fig. 1. Reduction from LCS to PMLG for $X = aababbbab$. The dashed edges are only shown from v_1 but similar edges are present from every v_i, $1 \leq i \leq |X|$. If $Y = baabbabaa$ then $P = \#^5b\#^5a\#^5a\#^5b\#^5b\#^5a\#^5b\#^5a\#^5a$.

Lemma 2. *The graph distance from v_i to v_j for any $j > i$ is at most $\lceil \log n \rceil$.*

Proof. Let i' be the largest value such that $i \leq i' \leq j$ and there exists edge $(v_i, v_{i'}) \in E$. By construction $i' = i + 2^x$ for some $x \geq 0$. We claim $i' > \frac{j-i}{2} + i$. Otherwise $i' = i + 2^x \leq \frac{j-i}{2} + i$ implies $i + 2^{x+1} \leq j$, contradicting that index i' was the largest possible. Since the distance between the current index and j can always be at least halved, by repeatedly apply the same process, we need at most $\lceil \log n \rceil$ additional edges before reaching j.

The correctness of the reduction is established by the following lemma.

Lemma 3. *There exists an LCS of length at least k for strings X and Y iff there exists a walk in G that matches P after at most $\delta = n - k$ substitutions to P.*

Proof. First assume there exists an LCS of length $k' \geq k$, with $X[i_1]$, $X[i_2]$,..., $X[i_{k'}]$ matching $Y[j_1]$, $Y[j_2]$,..., $Y[j_{k'}]$. We obtain a walk on G as follows: starting at vertex v_{i_1}, we traverse the self-loop (v_{i_1}, v_{i_1}) until we reach the $Y[j_1]$ in P, substituting symbols in P to # as necessary. Then we follow edge (v_{i_1}, u_{i_1}) matching $Y[j_1]$ in P. We now traverse the edge (u_{i_1}, v_{i_1+1}) and the shortest path from v_{i_1+1} to v_{i_2}, which by Lemma 2 has at most $\lceil \log n \rceil$ edges. We next traverse the self-loop (v_{i_2}, v_{i_2}) until reaching the symbol $Y[j_2]$ in P, at which point we match $Y[j_2]$ with the edge (v_{i_2}, u_{i_2}). This process is repeated until $Y[j_{k'}]$ is matched to the edge $(v_{i_{k'}}, u_{i_{k'}})$. If $i_{k'} = |X|$, the edge $(u_{|X|}, u_{|X|})$ is traversed for any remaining symbols in P. Exactly $n - k' \leq n - k = \delta$ symbols in P are substituted to #.

Next suppose there exists a walk in G that matches P with $\delta' \leq \delta$ substitutions. This implies that $n - \delta'$ of the non-#-symbols in P are not substituted and instead matched with symbols on edges (v_i, u_i). By construction, once an edge (v_i, u_i) is traversed, the next edge with a non-#-label traversed is an edge $(v_{i'}, u_{i'})$ where $i' > i$. Hence, the non-# symbols in P matched with edges in G correspond to a common subsequence of X and Y of length $n - \delta' \geq n - \delta = k$.

It can be easily shown that the statement of Lemma 3 holds when deletions and insertions are also allowed to P. Lemma 4 proves this result.

Lemma 4. *Given graph G and path P as in our reduction, if a walk minimizes the number of edits to P, we can assume only substitutions are made.*

Proof. Any substring of P consisting of only # can be matched without edit cost from any vertex v_i, $1 \leq i \leq |X|$. From vertices u_i, $1 \leq i \leq |X|$, an edge with # needs to be traversed regardless so it would be suboptimal to delete any # in P that could be matched on an edge (u_i, v_{i+1}). Combining these, an optimal solution never deletes a substring of P of the form $\#^x$, $x \geq 1$. This leaves only substrings that contain some symbol $Y[i]$. However, the cost for deleting any such substring is at least the cost of substituting $Y[i]$ to a #-symbol. We conclude that no deletions need to be made to P in an optimal solution.

For insertions, a similar argument holds. Any insertion of a substring of the form $\#^x$, $x \geq 1$, is clearly suboptimal since there exist enough #-symbols to traverse from any two vertices v_i and v_j. An insertion that includes a non-#-symbol is also unnecessary, since the edge matched against that symbol could have been not traversed for the same cost.

2.1 Hardness of PMLG over Binary Alphabet

Let $\Sigma' = \Sigma \cup \{\#\}$, $\sigma = |\Sigma'| \geq 3$, and $\ell = 2\lceil \log \sigma \rceil$. We will create our own constant weight binary code (i.e., one where all codewords have the same number of 1's) for Σ'. We first take $t = \lceil \log \sigma \rceil$. This makes $\binom{\ell}{t} \geq \sigma$ and allows us to assign to every symbol in Σ' a distinct binary string of length ℓ containing exactly t 1's. Let $\text{enc}(\alpha)$ denote this encoding for $\alpha \in \Sigma'$. Controlling the number of 1's allows us to compute the cost of an optimal solution, as described next. We modify the earlier reduction by replacing every edge (u, v) (allowing for $v = u$) having label $\alpha \in \Sigma'$ with:

- A directed path from u to v that matches $(10^{t-2+\ell})^t 0^{t-1}\,\text{enc}(\alpha)$. These paths are called *symbol paths*;
- A parallel directed path starting and ending at the same vertices (or vertex) that matches the string $(10^{t-2+\ell})^t 1^{t-1} 0^\ell$. These are called *escape paths.*

See Figure 2.

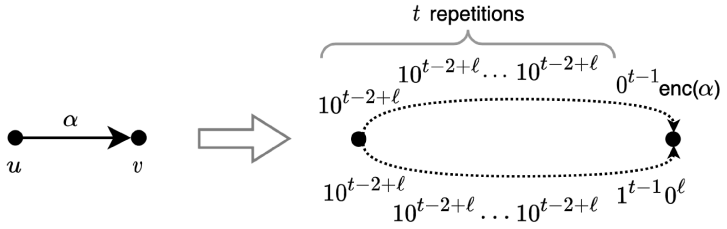

Fig. 2. Conversion of an edge in G with label α from u to v to two paths in G'.

We denote the resulting graph as G'. The pattern P' is created by replacing every symbol $P[i]$ with $(10^{t-2+\ell})^t 1^{t-1}\,\mathrm{enc}(P[i])$ for $1 \leq i \leq |P|$. Let

$$\begin{aligned}\delta' &= t(n-k) + (t-1)(|P| - (n-k)) \\ &= t(n-k) + (t-1)(n(\lceil \log n \rceil + 2) - (n-k)).\end{aligned}$$

Note that $|V'|$, $|E'|$, and $|P'|$ are $O(n \log n \cdot \log^2 \sigma)$.

Lemma 5. *Any walk in G' matching P' with at most δ' mismatches must start at some vertex corresponding to an original vertex in V.*

Proof. First, consider a walk starting at some vertex w internal to a subdivided path (i.e., one not corresponding to an original vertex in G) and where w does not have an edge labeled 1 leaving it. By construction, this causes substrings of the form 10^{t-2+l} in P to not be synchronized and creates at least t mismatches for every substring of the form $(10^{t-2+\ell})^t 1^{t-1}\,\mathrm{enc}(P[i])$. To see this, note that at most one of the 1's in the prefix $(10^{t-2+\ell})^t$ can be matched to an edge with label 1 and at least one of the 0's is mismatched to a 1 edge as well.

If the walk starts at a non-original vertex with an edge leaving it labeled 1 that is not internal to a subpath labeled 1^t in an escape path or a symbol encoding, then for each substring of the form $(10^{t-2+\ell})^t 1^{t-1}\,\mathrm{enc}(P[i])$ in P', the substring $\mathrm{enc}(\alpha)$ is forced to traverse a subpath labeled 0^t, causing at least t mismatches once again.

Finally, suppose the walk starts at a non-original vertex with an edge leaving it labeled 1, but internal to a subpath labeled 1^t in some escape path or a symbol encoding. Then for each substring $(10^{t-2+\ell})^t 1^{t-1}\,\mathrm{enc}(P[i])$ in P', the prefix $(10^{t-2+\ell})^t$ once again causes t mismatches from not being synchronized.

In all cases, at least $t|P| > \delta'$ mismatches are causes in total.

Lemma 6. *There exists an LCS of length at least k for strings X and Y iff there exists a walk in G' that matches P' after at most δ' substitutions to P'.*

Proof. First, suppose there exists an LCS of length at least k. Follow the walk in G' corresponding to the walk in G that requires at most δ substitutions to P. When doing so, take the symbol path when the symbol in P matched the corresponding edge in G, and the escape path otherwise. This incurs $t-1$ mismatches per subdivided edge corresponding to a match and t mismatches per subdivided edge corresponding to a mismatch. Hence the total number of mismatches is at most $t(n-k) + (t-1)(|P| - (n-k)) = \delta'$.

Next, suppose there exists a walk in G' matching P' with at most δ' mismatches. By Lemma 5, substrings of the form $(10^{t-2+\ell})^t 1^{t-1}\,\mathrm{enc}(P[i])$ are matched (after substitutions) to sub-paths in the walk that start at the beginning of symbol or escape paths. If $\mathrm{enc}(P[i]) \neq \mathrm{enc}(\alpha)$, then the number of mismatches for that substring is t since the number of mismatches for matching the escape path is t (Hamming distance of $1^{t-1}\,\mathrm{enc}(P[i])$ and $1^{t-1}0^\ell$), versus the symbol path, which is at least t (Hamming distance of $1^{t-1}\,\mathrm{enc}(P[i])$ and $0^{t-1}\,\mathrm{enc}(\alpha)$). If $\mathrm{enc}(P[i]) = \mathrm{enc}(\alpha)$, by matching the symbol path, the number of mismatches is

$t-1$ (Hamming distance of $1^{t-1}\,\mathrm{enc}(P[i])$ and $0^{t-1}\,\mathrm{enc}(\alpha)$). We conclude that in an optimal solution the total number of mismatches is t times the number of mismatched symbols between P and the corresponding walk in G, plus $t-1$ times the number of matched symbols between P and the corresponding walk in G. Hence, if the number mismatches for G' is at most $\delta' = t(n-k)+(t-1)(|P|-(n-k))$, the number mismatched symbols in G is at most $n-k=\delta$. By Lemma 3, this implies the LCS of X and Y is at least k.

This completes the proof of Theorem 1.

3 Quantum Algorithm for PMLG

We will use Durr et al.'s [11] single-source shortest path algorithm as a black box. Their algorithm is a modification of Dijkstra's algorithm that utilizes a minimum finding version of Grover's search to obtain a quantum time/query complexity of $\tilde{O}(\sqrt{|V||E|})$ on a graph $G=(V,E)$. It solves the st-shortest path problem correctly with constant probability greater than $\frac{1}{2}$. The version of this algorithm that we are using assumes the graph is represented using adjacency lists (in the form of an oracle). Given that the outdegree v_i is $d^+(v_i)$, the oracles $f_i : [d^+(v_i)] \mapsto \{1,...,|V|\} \times \mathbb{N}$ are defined as

$$f_i(j) = (j^{th} \text{ vertex adjacent to vertex } v_i, \text{ weight on the corresponding edge}).$$

We assume that $|E| = \Omega(|V|)$ and assign an arbitrary ordering to V.

A reduction from PMLG to the Single Source Shortest Path problem on an *alignment graph* was shown by Amir et al. in [5]. We will be using the oracles for G to implicitly construct an alignment graph for $G=(V,E)$ and $P[1,m]$, denoted G'. The number of vertices in G' is $\Theta(|V|m)$ and the number of edges is $\Theta(|E|m)$. Because of this, if we explicitly constructed the alignment graph, the $\Theta(|E|m)$ edges would result in no speed up over the classical algorithm. The key insight into efficiently using Durr et al.'s algorithm is that G' need not be explicitly constructed to simulate the oracles used by the shortest path algorithm. We show how the output of the oracles for G' can be computed in constant time given the oracles for G and P.

Our algorithm allows for substitutions, insertions, and deletions. Assume we have substitution cost S, deletion cost D, and insertion cost I. The alignment graph $G'=(V',E')$ is as follows: The vertex set is

$$V' = \{v_i^j \mid 1 \le i \le |V|, 1 \le j \le m+1\} \cup \{s,t\}.$$

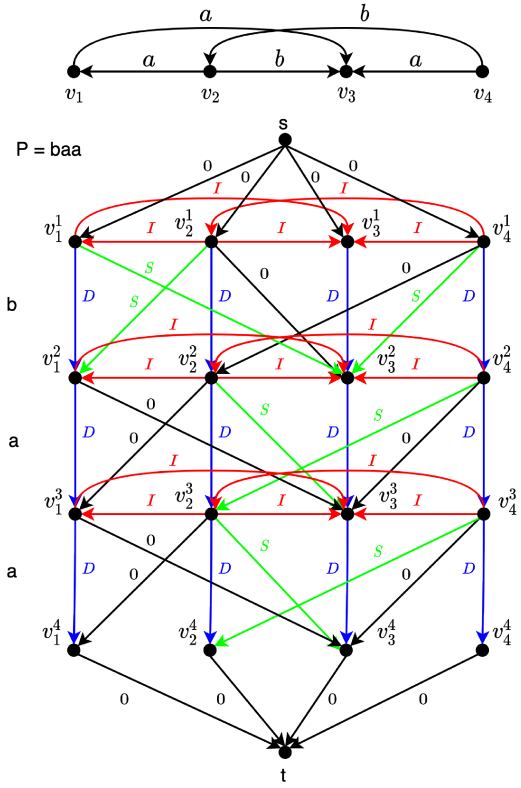

Fig. 3. An alignment graph G' (bottom) that is constructed from the starting graph G (top) and pattern $P = baa$. The edges labeled I correspond to insertion and have weight I; the edges labeled S correspond to substitution and have weight S; the edges labeled D correspond to deletion and have weight D; the black edges correspond to an exact match and have weight 0.

We denote edges with the triple (start-vertex, end-vertex, weight). The edge set for G' is

$$
\begin{aligned}
E' = &\{(s, v_i^1, 0) \mid 1 \le i \le |V|\} \ \cup \\
&\{(v_i^j, v_h^j, I) \mid 1 \le j \le m, (v_i, v_h) \in E\} \ \cup \\
&\{(v_i^j, v_h^{j+1}, 0) \mid 1 \le j \le m, (v_i, v_h) \in E \text{ and } label((v_i, v_h)) = P[j]\} \ \cup \\
&\{(v_i^j, v_h^{j+1}, S) \mid 1 \le j \le m, (v_i, v_h) \in E \text{ and } label((v_i, v_h)) \ne P[j]\} \ \cup \\
&\{(v_i^j, v_i^{j+1}, D) \mid 1 \le i \le |V|, 1 \le j \le m\} \ \cup \\
&\{(v_i^{m+1}, t, 0) \mid 1 \le i \le |V|\}.
\end{aligned}
$$

See Figure 3 for an example alignment graph.

The linearized index for s is 0, for t it is $(m+1)|V|+1$, and for v_i^j, $1 \leq i \leq |V|$, $1 \leq j \leq m+1$, it is $(j-1)|V|+i$.

For $1 \leq i \leq |V|$, $1 \leq j \leq m$, we have that $d^+(v_i^j) = 2d^+(v_i)+1$ and the oracle is $f_i^j : [d^+(v_i^j)] \mapsto \{0, ..., |V'|-1\} \times \mathbb{N}$, where

$$f_i^j(k) = \begin{cases} ((j-1)|V| + f_i(k), I) & 1 \leq k \leq d^+(v_i) \\ (j|V| + f_i(k - d^+(v_i)), 0) & d^+(v_i)+1 \leq k \leq 2d^+(v_i) \text{ and} \\ & \quad label((v_i, v_{f_i(k-d^+(v_i))})) = P[j] \\ (j|V| + f_i(k - d^+(v_i)), S) & d^+(v_i)+1 \leq k \leq 2d^+(v_i) \text{ and} \\ & \quad label((v_i, v_{f_i(k-d^+(v_i))})) \neq P[j] \\ (j|V| + i, D) & k = 2d^+(v_i)+1 \end{cases}$$

For $1 \leq i \leq |V|$ and $j = m+1$, $d^+(v_i^j) = 1$ and $f_i^j(1) = ((m+1)|V|+1, 0)$. For vertex s and $1 \leq k \leq |V|$, $f_0(k) = (k, 0)$.

Lemma 7 ([5]). *There exists an st-path in the alignment graph G' with total weight δ iff there exists a walk in G that P matches after δ edits.*

Applying the algorithm of Durr et al. and utilizing the oracles above gives an algorithm running in quantum time $\tilde{O}(\sqrt{|V'||E'|})$ for PMLG. Using that $|V'| = (m+1)|V| + 2$ and $|E'| = \Theta(m|E|)$ this has query/time complexity $\tilde{O}(m\sqrt{|V||E|})$.

Theorem 2. *There exists a quantum algorithm that solves PMLG (exact matching, matching with substitutions to P, or matching with substitutions, insertions, and deletions to P) with constant probability greater than $\frac{1}{2}$ and has $\tilde{O}(m\sqrt{|V||E|})$ quantum time and query complexity.*

4 Discussion

We leave open the problem of establishing the same reduction from LCS to PMLG when edits (substitutions, insertion, and deletions) are allowed to the pattern and the PMLG alphabet is binary. Lemma 4 establishes this result for polynomial sized alphabets. Note that the hardness of LCS under NC-QSETH (Lemma 1) holds for constant-sized alphabets, thus Corollary 2 can be extended to PMLG with edits to the pattern for constant-sized alphabets.

Our reduction from LCS creates a sparse graph. A subquadratic time reduction to a dense graph would give an improved quantum algorithm for LCS, suggesting the challenge of finding such a reduction. Moreover, the graph in our reduction is cyclic. This is interesting in light of improvements in the query complexity of quantum algorithms for recognizing if a string is in a regular language when the monoid representation of the regular language is acyclic [2]. If these results for monoids can be efficiently transferred to acyclic NFAs, it suggests the challenge of finding a reduction from LCS to PMLG on DAGs when $\delta = 0$.

Acknowledgement. This research is supported in part by the U.S. National Science Foundation (NSF) grants CCF-2146003 and CCF-2112643.

References

1. Pangaia (Nov 2020), https://www.pangenome.eu/
2. Aaronson, S., Grier, D., Schaeffer, L.: A quantum query complexity trichotomy for regular languages. In: 2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS). pp. 942–965. IEEE (2019)
3. Akmal, S., Jin, C.: Near-optimal quantum algorithms for string problems. In: Naor, J.S., Buchbinder, N. (eds.) Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022. pp. 2791–2832. SIAM (2022). https://doi.org/10.1137/1.9781611977073.109, https://doi.org/10.1137/1.9781611977073.109
4. Alanko, J., D'Agostino, G., Policriti, A., Prezza, N.: Regular languages meet prefix sorting. In: Chawla, S. (ed.) Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020. pp. 911–930. SIAM (2020). https://doi.org/10.1137/1.9781611975994.55, https://doi.org/10.1137/1.9781611975994.55
5. Amir, A., Lewenstein, M., Lewenstein, N.: Pattern matching in hypertext. J. Algorithms **35**(1), 82–99 (2000). https://doi.org/10.1006/jagm.1999.1063, https://doi.org/10.1006/jagm.1999.1063
6. Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H.: Elementary gates for quantum computation. Physical review A **52**(5), 3457 (1995)
7. Buhrman, H., Loff, B., Patro, S., Speelman, F.: Memory compression with quantum random-access gates. CoRR **abs/2203.05599** (2022). https://doi.org/10.48550/arXiv.2203.05599, https://doi.org/10.48550/arXiv.2203.05599
8. Buhrman, H., Patro, S., Speelman, F.: A framework of quantum strong exponential-time hypotheses. In: Bläser, M., Monmege, B. (eds.) 38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference). LIPIcs, vol. 187, pp. 19:1–19:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.STACS.2021.19, https://doi.org/10.4230/LIPIcs.STACS.2021.19
9. Chen, S., Krusche, P., Dolzhenko, E., Sherman, R.M., Petrovski, R., Schlesinger, F., Kirsche, M., Bentley, D.R., Schatz, M.C., Sedlazeck, F.J., et al.: Paragraph: a graph-based structural variant genotyper for short-read sequence data. Genome biology **20**(1), 1–13 (2019)
10. Consortium, C.P.G.: Computational pan-genomics: status, promises and challenges. Briefings in bioinformatics **19**(1), 118–135 (2018)
11. Dürr, C., Heiligman, M., Høyer, P., Mhalla, M.: Quantum query complexity of some graph problems. SIAM J. Comput. **35**(6), 1310–1328 (2006). https://doi.org/10.1137/050644719, https://doi.org/10.1137/050644719
12. Eggertsson, H.P., Kristmundsdottir, S., Beyter, D., Jonsson, H., Skuladottir, A., Hardarson, M.T., Gudbjartsson, D.F., Stefansson, K., Halldorsson, B.V., Melsted, P.: Graphtyper2 enables population-scale genotyping of structural variation using pangenome graphs. Nature communications **10**(1), 1–8 (2019)

13. Eizenga, J.M., Novak, A.M., Sibbesen, J.A., Heumos, S., Ghaffaari, A., Hickey, G., Chang, X., Seaman, J.D., Rounthwaite, R., Ebler, J., et al.: Pangenome graphs. Annual Review of Genomics and Human Genetics **21** (2020)
14. Equi, M., de Griend, A.M., Mäkinen, V.: From bit-parallelism to quantum: Breaking the quadratic barrier. CoRR **abs/2112.13005** (2021), https://arxiv.org/abs/2112.13005
15. Equi, M., Grossi, R., Mäkinen, V., Tomescu, A., et al.: On the complexity of string matching for graphs. In: 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
16. Gagie, T., Manzini, G., Sirén, J.: Wheeler graphs: A framework for bwt-based data structures. Theor. Comput. Sci. **698**, 67–78 (2017). https://doi.org/10.1016/j.tcs.2017.06.016, https://doi.org/10.1016/j.tcs.2017.06.016
17. Garrison, E., Sirén, J., Novak, A.M., Hickey, G., Eizenga, J.M., Dawson, E.T., Jones, W., Garg, S., Markello, C., Lin, M.F., et al.: Variation graph toolkit improves read mapping by representing genetic variation in the reference. Nature biotechnology **36**(9), 875–879 (2018)
18. Gibney, D., Hoppenworth, G., Thankachan, S.V.: Simple reductions from formula-sat to pattern matching on labeled graphs and subtree isomorphism. In: Le, H.V., King, V. (eds.) 4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021. pp. 232–242. SIAM (2021). https://doi.org/10.1137/1.9781611976496.26, https://doi.org/10.1137/1.9781611976496.26
19. Gibney, D., Thankachan, S.V.: On the hardness and inapproximability of recognizing wheeler graphs. In: Bender, M.A., Svensson, O., Herman, G. (eds.) 27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany. LIPIcs, vol. 144, pp. 51:1–51:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPIcs.ESA.2019.51, https://doi.org/10.4230/LIPIcs.ESA.2019.51
20. Gibney, D., Thankachan, S.V.: On the complexity of recognizing wheeler graphs. Algorithmica **84**(3), 784–814 (2022). https://doi.org/10.1007/s00453-021-00917-5, https://doi.org/10.1007/s00453-021-00917-5
21. Gibney, D., Thankachan, S.V., Aluru, S.: The complexity of approximate pattern matching on de bruijn graphs. In: Pe'er, I. (ed.) Research in Computational Molecular Biology - 26th Annual International Conference, RECOMB 2022, San Diego, CA, USA, May 22-25, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13278, pp. 263–278. Springer (2022). https://doi.org/10.1007/978-3-031-04749-7_16, https://doi.org/10.1007/978-3-031-04749-7_16
22. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 212–219 (1996)
23. Gruska, J., et al.: Quantum computing, vol. 2005. McGraw-Hill London (1999)
24. Hariharan, R., Vinay, V.: String matching in õ(sqrt(n)+sqrt(m)) quantum time. J. Discrete Algorithms **1**(1), 103–110 (2003). https://doi.org/10.1016/S1570-8667(03)00010-8, https://doi.org/10.1016/S1570-8667(03)00010-8
25. Hickey, G., Heller, D., Monlong, J., Sibbesen, J.A., Sirén, J., Eizenga, J., Dawson, E.T., Garrison, E., Novak, A.M., Paten, B.: Genotyping structural variants in pangenome graphs using the vg toolkit. Genome biology **21**(1), 1–17 (2020)

26. Jain, C., Zhang, H., Gao, Y., Aluru, S.: On the complexity of sequence-to-graph alignment. Journal of Computational Biology **27**(4), 640–654 (2020)
27. Li, H., Feng, X., Chu, C.: The design and construction of reference pangenome graphs with minigraph. Genome biology **21**(1), 1–19 (2020)
28. Limasset, A., Flot, J.F., Peterlongo, P.: Toward perfect reads: self-correction of short reads via mapping on de bruijn graphs. Bioinformatics **36**(5), 1374–1381 (2020)
29. Magniez, F., Nayak, A., Roland, J., Santha, M.: Search via quantum walk. SIAM journal on computing **40**(1), 142–164 (2011)
30. Manber, U., Wu, S.: Approximate string matching with arbitrary costs for text and hypertext. In: Advances In Structural And Syntactic Pattern Recognition, pp. 22–33. World Scientific (1992)
31. Morisse, P., Lecroq, T., Lefebvre, A.: Hybrid correction of highly noisy long reads using a variable-order de bruijn graph. Bioinformatics **34**(24), 4213–4222 (2018)
32. Navarro, G.: Improved approximate pattern matching on hypertext. Theor. Comput. Sci. **237**(1-2), 455–463 (2000). https://doi.org/10.1016/S0304-3975(99)00333-3, https://doi.org/10.1016/S0304-3975(99)00333-3
33. Park, K., Kim, D.K.: String matching in hypertext. In: Annual Symposium on Combinatorial Pattern Matching. pp. 318–329. Springer (1995)
34. Paten, B., Novak, A.M., Eizenga, J.M., Garrison, E.: Genome graphs and the evolution of genome inference. Genome research **27**(5), 665–676 (2017)
35. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994. pp. 124–134. IEEE Computer Society (1994). https://doi.org/10.1109/SFCS.1994.365700, https://doi.org/10.1109/SFCS.1994.365700
36. Tzanis, E.: A quantum algorithm for string matching. In: Guimarães, N., Isaías, P.T. (eds.) AC 2005, Proceedings of the IADIS International Conference on Applied Computing, Algarve, Portugal, February 22-25, 2005, Volume 2. pp. 374–377. IADIS (2005)