Quantum Algorithm for the Longest Trail Problem

Kamil Khadiev and Ruslan Kapralov

Kazan Federal University, Kazan, Russia
email: kamilhadi@gmail.com

Abstract

We present the quantum algorithm for the Longest Trail Problem. The problem is to search the longest edge-simple path for a graph with n vertexes and m edges. Here edge-simple means no edge occurs in the path twice, but vertexes can occur several times. The running time of our algorithm is \(O^*(1.728^m)\).

1 Introduction

Quantum computing [17, 2, 1] is one of the hot topics in computer science of the last decades. There are many problems where quantum algorithms outperform the best-known classical algorithms. Some of them can be founded here [8, 13]. Problems for graphs are examples of such problems [15, 14, 3, 10]. One of the most important performance metrics in this regard is query complexity; and we investigate problems using this metric for complexity.

In this paper, we consider the Longest Trail Problem (LTP). The problem is the following one. Let us consider a graph with n vertexes and m edges. The problem is to search the longest edge-simple path. Here edge-simple means no edge occurs in the path twice, but vertexes can occur several times. The longest means the path has the maximal possible number of edges.

The problem is strongly related to the longest path problem (LPP) that is searching the longest vertex-simple path. Here vertex-simple means no vertex occurs in the path twice.

There are many practical applications of these problems, for example, [4, 18]. Both problems are NP-hard [16]. The NP-hardness of LTP problem was discussed in [7].

The simple classical solution for the problem can be a brute force algorithm that checks all possible paths and searching the required one. Such solution works in \(O(m!) = O(m^m)\) running time. This solution can be used as a base of a quantum algorithm because the classical algorithm solves a search problem. Therefore, we can use Grover Search algorithm [11, 6] and obtain a quantum algorithm that works in \(O(\sqrt{m}) = O(m^{0.5m})\). At the same time, there is a better classical algorithm that is based on the Dynamic programming approach [5, 12]. This classical algorithm for the LTP problem works in \(O^*(2^m)\) running time, where \(O^*\) hides polylog factors. The algorithm is not a simple search algorithm. That is why we cannot directly use the Grover Search algorithm for quantum speed-up, and we cannot obtain a complexity \(O^*(1.4^m)\) using this way. We present a quantum algorithm that works in \(O^*(1.728^m)\) running time. The algorithm is based on Grover Search algorithm [11, 6], quantum minimum finding algorithm [9, 10] and quantum ideas for dynamic programming on Boolean cube [3].

The structure of the paper is the following. Section 2 contains preliminaries. Then, we discuss algorithm in Section 3.

2 Preliminaries

2.1 The Longest Trail Problem

Let \(G = (V, E)\) be an unweighted, underacted graph, where \(V\) is a set of vertexes, and \(E\) is a set of edges. Let \(m = |E|\) be the number of edges and \(n = |V|\) be the number of vertexes.

Let a sequence of edges \(P = (e_1, \ldots, e_{\ell})\) be a path if each sequentially pair of edges \(e_i\) and \(e_{i+1}\) has common vertex, for \(j \in \{1, \ldots, \ell - 1\}\). A path is edge-simple if the sequence has no duplicates i.e.,
for any $j \neq j'$ we have $e_{ij} \neq e_{ij'}$. Let $|P| = \ell$ be the length of a path $P$. Let $\mathcal{P}(G)$ be the set of all possible paths for a graph $G$.

The problem is to the longest path i.e., any path $P_{long}$ such that $|P_{long}| = \max \{|P| : P \in \mathcal{P}(G)\}$.

2.2 Quantum Query Model

We use the standard form of the quantum query model. Let $f : D \rightarrow \{0,1\}, D \subseteq \{0,1\}^N$ be an $N$ variable function. An input for the function is $x = (x_1,\ldots,x_N) \in D$ where $x_i \in \{0,1\}$ for $i \in \{1,\ldots,N\}$.

We are given oracle access to the input $x$, i.e. it is realized by a specific unitary transformation usually defined as $|i\rangle|z\rangle|w\rangle \rightarrow |i\rangle|z + x_i \text{ (mod 2)}\rangle|w\rangle$ where the $|i\rangle$ register indicates the index of the variable we are querying, $|z\rangle$ is the output register, and $|w\rangle$ is some auxiliary work-space. It can be interpreted as a sequence of control-not transformations such that we apply inversion operation (X-gate) to the second register that contains $|z\rangle$ in a case of the first register equals $i$ and the variable $x_i = 1$. We interpret the oracle access transformation as $N$ such controlled transformations for each $i \in \{1,\ldots,N\}$.

An algorithm in the query model consists of alternating applications of arbitrary unitaries independent of the input and the query unitary, and a measurement in the end. The smallest number of queries for an algorithm that outputs $f(x)$ with a probability that is at least $\frac{2}{3}$ on all $x$ is called the quantum query complexity of the function $f$ and is denoted by $Q(f)$. We refer the readers to [17, 2, 1] for more details on quantum computing.

In this paper’s quantum algorithms, we refer to the quantum query complexity as the quantum running time. We use modifications of Grover’s search algorithm [11, 6] as quantum subroutines. For these subroutines, time complexity is more than query complexity for additional log factor.

3 Algorithm

We discuss our algorithm in this section. Let us consider a function $L : 2^E \times E \times E \rightarrow \mathbb{R}$ where $2^E$ is the set of all subsets of $E$. The function $L$ is such that $L(S, v, u)$ is the length of the longest path that uses only edges from the set $S$, starts from the edge $v$, and finishes in the edge $u$.

Let the function $F : 2^E \times E \times E \rightarrow E^*$ be such that $F(S, v, u)$ is the longest path that uses only edges from the set $S$, starts from the edge $v$, and finishes in the edge $u$.

It is easy to see that $L(\{v\}, v, v) = 1$ and $F(\{v\}, v, v) = (v)$ for any $v \in E$ because the set has only one edge and it is the only path in the set.

Another property of these functions is

**Property 1** Suppose $S \in 2^E, v, u \in E$, an integer $k \leq |S|$. The function $L$ is such that

$$L(S, v, u) = \max_{S' \subset S, |S'| = k, y \in S'} (L(S', v, y) + L((S')\cup \{y\}, y, u))$$

and $F(S, u, v)$ is the path that is concatenation of corresponding paths.

**Proof.** Let $P^1 = F(S', v, y)$ and $P^2 = F((S\setminus S')\cup \{y\}, y, u)$. The path $P = P^1 \circ P^2$ belongs to $S'$, starts from $v$ and finishes in $u$, where $\circ$ means concatenation of paths excluding the duplication of common edge $y$. Because of definition of $L$, we have $L(S, v, u) \geq |P|$. Assume that there is $T = (e_1,\ldots,e_\ell)$ such that $\ell = |T| = L(S, v, u)$ and $|T| > |P|$. Let us select $S''$ such that $|S''| = k, S'' \subset S$ and there is $j < |T|$ such that $R^1 = e_1,\ldots,e_j \in S''$ and $R^2 = e_j,\ldots,e_{\ell} \notin S''\setminus \{e_j\}$. Then $|R^1| \leq |P^1|$ and $|R^2| \leq |P^2|$ by definition of $F$ and $L$. Therefore, $|R| = |R^1| + |R^2| - 1 \leq |P^1| + |P^2| - 1 = |P|$. We obtain a contradiction with assumption. \qed

As a corollary we obtain the following result:

**Corollary 1** Suppose $S \in 2^E, v, u \in E$, $I(u)$ is the set of all edges that has common vertex with $u$. The function $L$ is such that

$$L(S, v, u) = \max_{y \in S \setminus \{v\}, y \in I(u)} (L(S\setminus \{u\}, v, y) + 1)$$

and $F(S, u, v)$ is the path that is the corresponding path.
Using this idea, we construct the following algorithm.

**Step 1.** Let $\alpha = 0.055$. We classically compute $L(s, v, u)$ and $F(s, v, u)$ for $|S| = (1 - \alpha)\frac{m}{4}$ and $v, u \in E$.

**Step 2.** Let $E_4 \subset E$ be such that $|E_4| = \frac{m}{4}$. Then, we have

$$L(E_4, u, v) = \max_{E_4 \subset E, |E_4| = m/4, y \in E_4} (L(E_4, v, y) + L((E_4 \setminus E_4) \cup \{y\}, y, u)).$$

Let $E_2 \subset E$ be such that $|E_2| = \frac{m}{2}$. Then, we have

$$L(E_2, u, v) = \max_{E_2 \subset E, |E_2| = m/2, y \in E_2} (L(E_2, v, y) + L((E_2 \setminus E_2) \cup \{y\}, y, u)).$$

Finally,

$$L(E, u, v) = \max_{E \subset E, |E| = m/2, y \in E} (L(E, v, y) + L((E \setminus E) \cup \{y\}, y, u)).$$

We can compute $L(E, u, v)$ and corresponding $F(E, u, v)$ using three nested procedures for maximum finding. As such procedure, we use Durr-Hoyer [9, 10] quantum minimum finding algorithm.

Note that the error probability for the Durr-Hoyer algorithm is at most 0.1. So, we use the standard boosting technique to decrease the total error probability to constant by $O(m)$ repetition of the maximum finding algorithm in each level.

Let us present the implementation of Step 1. Assume that $\mathcal{I}(u)$ is the sequence of edges that have a common vertex with the edge $u$. Let us present a recursive function $\text{GetLen}(S, v, u)$ for $S \subset 2^E, u, v \in E$ with caching that is Dynamic Programming approach in fact. The function is based on Corollary 1.

**Algorithm 1 GetLen($S, v, u$).**

```plaintext
if $v = u$ and $S = \{v\}$ then  
  $L(\{v\}, v, v) \leftarrow 1$
  $F(\{v\}, v, v) \leftarrow (v)
end if

if $L(S, v, u)$ is not computed then
  $len \leftarrow 1$
  $path \leftarrow ()$
  for $y \in \mathcal{I}(u)$ do
    if $y \in S \setminus \{u\}$ and $\text{GetLen}(S \setminus \{u\}, v, y) + 1 > len$ then
      $len \leftarrow L(S \setminus \{u\}, v, y) + 1$
      $path \leftarrow F(S \setminus \{u\}, v, y) \cup u$
    end if
  end for
  $L(S, v, u) \leftarrow len$
  $F(S, v, u) \leftarrow path$
end if
return $L(S, v, u)$
```

**Algorithm 2 Step1.**

```plaintext
for $S \subset 2^E$ such that $|S| = (1 - \alpha)\frac{m}{4}$ do
  for $v \in E$ do
    for $u \in E$ do
      if $v \in S$ and $u \in S$ then
        $\text{GetLen}(S, v, u)$: We are computing $L(S, v, u)$ and $F(S, v, u)$ but we are not needing this results at the moment. We need it for Step 2.
      end if
    end for
  end for
end for
```
Let $\text{QMAX}(\alpha_1, \ldots, \alpha_N)$ be the implementation of the quantum maximum finding algorithm [9] for a sequence $x_1, \ldots, x_N$.

The most nested quantum maximum finding algorithm for some $E_4 \subset E$, $|E_4| = \frac{m}{4}$ and $u, v \in E$ is

$$\text{QMAX}((L(E_4, v, y) + L(E_4, u, y) : E_4 \subset E_4, |E_4| = (1 - \alpha) \frac{m}{4}, y \in E_4))$$

The middle quantum maximum finding algorithm for some $E_2 \subset E$, $|E_2| = \frac{m}{2}$ and $u, v \in E$ is

$$\text{QMAX}((L(E_4, v, y) + L(E_4, \setminus \{y\}, y, u) : E_4 \subset E_2, |E_4| = \frac{n}{4}, y \in E_4))$$

Note that $|E_4| = m/4$ and $|E_2| = m/4$. We use the invocation of QMAX (the most nested quantum maximum finding algorithm) instead of $L(E_4, v, y)$ and $L(E_2 \setminus \{y\}, y, u)$.

The final quantum maximum finding algorithm for some $u, v \in E$ is

$$\text{QMAX}((L(E_2, v, y) + L(E_2 \setminus \{y\}, y, u) : E_2 \subset E, |E_2| = \frac{m}{2}, y \in E_2))$$

Note that $|E_2| = m/2$ and $|E_2 \setminus \{y\}| = m/2$. We use the invocation of QMAX (the middle quantum maximum finding algorithm) instead of $L(E_2, v, y)$ and $L((E_2 \setminus \{y\}) \cup \{y\}, y, u)$.

The procedure QMAX returns not only the maximal value, but the index of the target element. Therefore, by the “index” we can obtain the target paths using $F$ function. So the result path is $P = P^1 \circ P^2$, where $P^1$ is the result path for $L(E_2, v, y)$ and $P^2$ is the result path for $L((E_2 \setminus \{y\}) \cup \{y\}, y, u)$.

$P^1 = P^{1,1} \circ P^{1,2}$, where $P^{1,1}$ is the result path for $L(E_4, v, y)$ and $P^{1,2}$ is the result path for $L((E_2 \setminus \{y\}) \cup \{y\}, y, u)$. By the same way we can construct $P^2 = P^{2,1} \circ P^{2,2}$.

$P^{1,1} = P^{1,1,1} \circ P^{1,1,2}$, where $P^{1,1,1}$ is the result path for $L(E_4, v, y)$ and $P^{1,1,2}$ is the result path for $L((E_4 \setminus \alpha_n) \cup \{y\}, y, u)$. Note, that this values were precomputed classically, and were stored in $F(E_4, v, y)$ and $F((E_4 \setminus \alpha_n) \cup \{y\}, y, u)$ respectively.

By the same way we can construct

$$P^{1,2} = P^{1,2,1} \circ P^{1,2,2},$$
$$P^{2,1} = P^{2,1,1} \circ P^{2,1,2},$$
$$P^{2,2} = P^{2,2,1} \circ P^{2,2,2}.$$ 

The final Path is

$$P = P^1 \circ P^2 = (P^{1,1} \circ P^{1,2}) \circ (P^{2,1} \circ P^{2,2}) =$$
$$((P^{1,1,1} \circ P^{1,1,2}) \circ (P^{1,2,1} \circ P^{1,2,2})) \circ ((P^{2,1,1} \circ P^{2,1,2}) \circ (P^{2,2,1} \circ P^{2,2,2}))$$

Let us present the final algorithm as Algorithm 3.

**Algorithm 3** Algorithm for LTP.

```plaintext
STEP1()
len ← −1
path ← ()
for v ∈ E do
    for u ∈ E do
        currentLen ← QMAX((L(E_2, v, y) + L((E_2 \setminus \{y\}) \cup \{y\}, y, u) : E_2 \subset E, |E_2| = m/2, y \in E_2))
        if len < currentLen then
            len ← currentLen
            path ← ((P^{1,1,1} \circ P^{1,1,2}) \circ (P^{1,2,1} \circ P^{1,2,2})) \circ ((P^{2,1,1} \circ P^{2,1,2}) \circ (P^{2,2,1} \circ P^{2,2,2}))
        end if
    end for
end for
return path
```

The complexity of the algorithm is presented in the next theorem.

**Theorem 1** Algorithm 3 solves LTP with $O^*(1.728^m)$ running time and constant bounded error.
Proof. The correctness of the algorithm follows from the above discussion. Let us present an analysis of running time.

Complexity of Step 1 (Classical preprocessing) is

\[ O^* \left( \left( \frac{m}{(1 - \alpha) \frac{2}{4}} \right) \right) = O^*(1.728^m). \]

Complexity of Step 2 (Quantum part) is complexity of three nested Durr-Hoyer maximum finding algorithms. Due to \([9, 11, 10, 1]\), the complexity is

\[ O^* \left( \sqrt{\left( \frac{m}{m/2} \right)} \cdot \sqrt{\left( \frac{m/2}{m/4} \right)} \cdot \sqrt{\left( \frac{m/4}{\alpha m/4} \right)} \right) = O^*(1.728^m). \]

We invoke Step 1 and Step 2 sequentially. Therefore the total complexity is the sum of complexities for these steps. So, the total complexity is \(O^*(1.728^m)\).

Only Step 2 has an error probability. The most nested invocation of the Durr-Hoyer algorithm has an error probability 0.1. Let us repeat it \(2m\) times and choose the maximal value among all invocations. The algorithm has an error only if all invocations have an error. Therefore, the error probability is \(0.1^{2m} = 100^{-m}\).

Let us consider the middle Durr-Hoyer algorithm’s invocation. The probability of success is the probability of correctness of maximum finding and the probability of input correctness, i.e., the correctness of all the nested Durr-Hoyer algorithm’s invocations. It is

\[ 0.9 \cdot (1 - 100^{-m})^\gamma, \text{ where } \gamma = \left( \frac{m/2}{m/4} \right), \]

\[ \geq 0.8, \text{ for enough big } m. \]

So, the error probability is at most 0.2.

Let us repeat the middle Durr-Hoyer algorithm \(2m\) times and choose the maximal value among all invocations. Similar to the previous analysis, the error probability is \(0.2^{2m} = 25^{-m}\).

Therefore, the total success probability that is the final Durr-Hoyer algorithm’s success probability is the following one.

\[ 0.9 \cdot (1 - 25^{-m})^\beta, \text{ where } \beta = \left( \frac{m}{m/2} \right), \]

\[ > 0.8, \text{ for enough big } m. \]

Therefore, the total error probability is at most 0.2. \(\square\)

Acknowledgments

The research is funded by the subsidy allocated to Kazan Federal University for the state assignment in the sphere of scientific activities, project No. 0671-2020-0065.

References

[1] F. Ablayev, M. Ablayev, J. Z. Huang, K. Khadiev, N. Salikhova, and D. Wu. On quantum methods for machine learning problems part i: Quantum tools. Big Data Mining and Analytics, 3(1):41–55, 2019.

[2] A. Ambainis. Understanding quantum algorithms via query complexity. In Proc. Int. Conf. of Math. 2018, volume 4, pages 3283–3304, 2018.

[3] Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 1783–1793. SIAM, 2019.
[4] David L. Applegate, Robert E. Bixby, Vasek C'vatal, and William J. Cook. *The traveling salesman problem: a computational study*. Princeton University Press, 2007.

[5] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM (JACM)*, 9(1):61–63, 1962.

[6] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, 1998.

[7] Marzio De Biasi. Is the longest trail problem easier than the longest path problem?, 2014. Stack Exchange. Theoretical Computer Science. https://cstheory.stackexchange.com/questions/20682/is-the-longest-trail-problem-easier-than-the-longest-path-problem.

[8] Ronald de Wolf. *Quantum computing and communication complexity*. 2001.

[9] C. Dürr and P. Høyer. A quantum algorithm for finding the minimum. *arXiv:quant-ph/9607014*, 1996.

[10] Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. In *International Colloquium on Automata, Languages, and Programming*, pages 481–493. Springer, 2004.

[11] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[12] Michael Held and Richard M Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied mathematics*, 10(1):196–210, 1962.

[13] Stephen Jordan. Quantum algorithms zoo, 2021. http://quantumalgorithmzoo.org/.

[14] K. Khadiev, D. Kravchenko, and D. Serov. On the quantum and classical complexity of solving subtraction games. In *Proceedings of CSR 2019*, volume 11532 of *LNCS*, pages 228–236. 2019.

[15] K. Khadiev and L. Safina. Quantum algorithm for dynamic programming approach for dags. applications for zhegalkin polynomial evaluation and some problems on dags. In *Proceedings of UCNC 2019*, volume 4362 of *LNCS*, pages 150–163. 2019.

[16] Eugene L Lawler. *Combinatorial optimization: networks and matroids*. Courier Corporation, 2001.

[17] M. A Nielsen and I. L Chuang. *Quantum computation and quantum information*. Cambridge univ. press, 2010.

[18] Jacob Scott, Trey Ideker, Richard M Karp, and Roded Sharan. Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology*, 13(2):133–144, 2006.