A Quantum Algorithm for Finding k-Minima

Kohei Miyamoto[1] * Masakazu Iwamura[1] † Koichi Kise[1] ‡

[1] *Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University*

Abstract. We propose a new *finding k-minima* algorithm and prove that the query complexity is $\mathcal{O}(\sqrt{kN})$, where N is the number of data indices. The primary difficulty of the problem is that it requires to return k answers. For the problem, an extension of the Amplitude Amplification (we call it *searching all marked k indices* algorithm) finds all k data with the query complexity of $\mathcal{O}(\sqrt{kN})$ if an appropriate threshold is given. We give a quantum algorithm that searches a good threshold with the complexity of $\mathcal{O}(\sqrt{N}\log k)$. In addition, we briefly prove the query complexity of the *searching all marked k-indices* algorithm, which is not well discussed so far. Our algorithm can be directly adapted to distance-related problems like k-Nearest Neighbor Search, clustering and classification.

The full version of the paper is followed by the extended abstract.

Keywords: Amplitude Amplification, Finding Minimum, Finding k-Minima

1 Introduction

Finding k-minima, which finds the k smallest data from N data indices, is an important problem, as it can be applied to k-nearest neighbor search like [1] and other quantum machine learning methods such as clustering and classification [2–5]. These machine learning methods are very important for analyzing big data. If the power of the quantum computer adapts to big data, big data that is hard to calculate on the classical computer can be analyzed.

The primary difficulty of the problem is that it requires to return k answers. Many quantum algorithms do not directly return multiple answers because the measurement of quantum states collapses the state of superposition. In a naive way, many trials are required to obtain all the results, which increases the computational cost linearly for the number of results. That is, $\mathcal{O}(k)$ trials are required for k results. It spoils the advantage of quantum algorithms.

In the problem, Dürr, et al. have proposed an $\mathcal{O}(\sqrt{kN})$ quantum algorithm for finding k-minima [6], where N is the number of data indices. However, it has some drawbacks. As the algorithm is developed for some graph problems, it is complex to adapt for the finding k-minima problem. In addition, a part of the algorithm is uncertain.

Hence, we propose a new *finding k-minima* algorithm and prove that the query complexity is $\mathcal{O}(\sqrt{kN})$. Though the proposed algorithm has the same query complexity as the one proposed by Dürr et al. [6], the proposed is free from the issues the existing method has. The proposed algorithm is based on following three algorithms: finding minimum (FM) algorithm [7], quantum counting (QC) algorithm [8, 9] and Amplitude Amplification (AA) [9–11]. The main idea of our algorithm is to find a good threshold for indices and find all indices that have less values than the value of the threshold index. FM algorithm and QC algorithm are used to find a good threshold index, and an extension of AA (we call this *searching all marked k-indices* algorithm) is used to find all k indices whose values are less than the value of the threshold index.

In addition, we re-formulate FM algorithm and *finding k-minima* algorithm following the manner of AA. Therefore, all of them can be compared more easily and clearly. AA is a quantum database search algorithm and is a generalization of Grover's algorithm [11]. From N indices, AA searches one of k indices that satisfy some certain condition with the query complexity of $\mathcal{O}(\sqrt{N/k})$. By using this AA, FM algorithm finds the minimum data from N data with the query complexity $\mathcal{O}(\sqrt{N})$. This complexity is smaller than that of the linear search in the classical computer ($\mathcal{O}(N)$).

2 Preliminaries

2.1 Amplitude Amplification (AA)

As AA treats indices of data instead of data themselves, we define the following.

Indices Let D be the set of all indices of data, where $|D| = N$.

Oracle Let $f(x)$ be a function that returns 1 or 0.

Marked indices A set of indices that satisfy $f(x) = 1$ is sometimes called marked indices. That is, $\{x \in D \mid f(x) = 1\}$.

Mapping function Since AA treats an index, we need a mean to access its value. Let $g(x)$ be a function that maps index x to the corresponding value.

2.2 Finding Minimum Algorithm

FM algorithm is a simple application of AA and helps understand the succeeding algorithms. This algorithm finds the minimum from D with the complexity of $\mathcal{O}(\sqrt{N})$ by iteratively updating threshold index t by

*miyamotokohei@protonmail.com
†masa@cs.osakafu-u.ac.jp
‡kise@cs.osakafu-u.ac.jp

oracle f_t. Oracle $f_t(x)$ is a function that marks index x such that $g(x) < g(t)$. That is,

$$f_t(x) = \begin{cases} 1, & \text{if } g(x) < g(t), \\ 0, & \text{if } g(x) \geq g(t). \end{cases} \quad (1)$$

The oracle marks the indices that have smaller values than the value of threshold t. Setting a marked index as the new threshold, the value of threshold decreases. Eventually, we obtain the index that has the minimum value.

In summary, FM algorithm is given as follows.

1. Select threshold index t from D uniformly at random.
2. Repeat the following process more than
$$22.5\sqrt{N} + 1.4\log^2 N$$
times.
   (a) Find index x such that $f_t(x) = 1$.
   (b) Set the found index x as threshold index t.
3. Return t as the index that has the minimum value in D.

2.3 Searching All Marked k-Indices Algorithm

The problem to search all marked k-indices from D and its query complexity are briefly discussed in [3, 12–15]. Here, we will give the complexity of the problem in an easy-to-understand way.

AA searches one of the k indices from D in $\mathcal{O}(\sqrt{N/k})$ query complexity [9–11]. By extending this, we can search all marked k indices by the following algorithm.

1. Initialize the set of search indices $M = \{x | x \in D, f(x) = 1\}$.
2. Initialize the set of already found indices $I = \emptyset$.
3. Repeat the following process until all k marked indices are found (which requires $\mathcal{O}(k)$ times).
   (a) Search index $m \in M$ from D by using AA with oracle
   $$f'(x) = \begin{cases} 1, & \text{if } f(x) = 1 \text{ and } x \notin I, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$
   (b) Add a found index m to I.
   (c) Remove m from M.

In the i-th iteration of this algorithm, as $|M| = k - i$, the query complexity to search a marked index is $\mathcal{O}(\sqrt{\frac{N}{k-i}})$. Therefore, the total query complexity is given as

$$\mathcal{O}\left(\sqrt{\frac{N}{k}} + \sqrt{\frac{N}{k-1}} + \cdots + \sqrt{N}\right) = \mathcal{O}(\sqrt{kN}), \quad (3)$$

because

$$\sum_{t=1}^{k} \sqrt{\frac{1}{i}} < 1 + \int_1^k \sqrt{\frac{1}{i}} di = 2\sqrt{k} - 1. \quad (4)$$

3 Conventional Finding k-Minima Algorithm

Dürr and Høyer have proposed an $\mathcal{O}(\sqrt{kN})$ algorithm that finds the k smallest indices *of different types* from D [6]. As the algorithm considers *types*, it is complex. Hence, we present it in an easier-to-understand way.

As it treats types, we consider the following conditions.

- Each index has a type.
- Let C_i be a set of indices that have type i.
- Let c be the number of types.

Dürr and Høyer's algorithm finds the indices of the c smallest values each of which is the minimum in each type. This means that so as to use the conventional algorithm for the general *finding k-minima* problem, all indices must be of different types, which corresponds to $c = N = |D|$. Hereafter, we assume all indices are of different types.

Intuitive explanation of the algorithm is that multiple thresholds are maintained while a single threshold is maintained in FM algorithm. Let T be a set of k thresholds and let $f_T(x)$ be an oracle function such that

$$f_T(x) = \begin{cases} 1, & \text{if } g(x) < g(t) \text{ for some } t \in T, \\ 0, & \text{if } g(x) \geq g(t) \text{ for some } t \in T. \end{cases} \quad (5)$$

Similar to Eq. (1) of FM algorithm, Eq. (5) is regarded as AA for *some* threshold t. However, the meaning of *some* is not clearly mentioned in [6]. Though the algorithm does not work well in the worst case, which selects the threshold index t that minimizes $g(t)$, it works in the following cases.

1. t is randomly chosen from T.
2. t is selected so as to maximize $g(t)$.

In the case of 2, which is the best case, such t is obtained by *finding maximum* algorithm [16]. Hence, its computational burden is $\mathcal{O}(\sqrt{k})$. It can be ignored, as it is small enough compared to the complexity of the whole algorithm (i.e., $\mathcal{O}(\sqrt{kN})$).

In addition, we have to keep the elements of T without duplication. So as to do that, the algorithm requires duplication check or removing T from search indices set

$$M = \{x \mid f_T(x) = 1, x \in D\}. \quad (6)$$

We assume that it removes T from M because the proposed method also removes already found indices from M.

By ignoring types of data, the conventional *finding k-minima* algorithm based on Dürr and Høyer's algorithm is given as follows.

1. Initialize set T as randomly chosen k indices from D.
2. Repeat the following forever.

(a) Randomly select a threshold index t from T.

(b) Find index x such that $f_T(x) = 1$ by AA.

(c) Find $t_{\max}$ such that $t_{\max} = \underset{t \in T}{\operatorname{argmax}}\, g(t)$ by *finding maximum* algorithm [16].

(d) Replace threshold index $t_{\max}$ with x.

In this algorithm, T is updated in a step-by-step manner and each updating step replaces the index that has the maximum value in T with the found index by AA. This is a kind of a greedy algorithm.

4 Proposed Finding k-Minima Algorithm

We propose a new *finding k-minima* algorithm with the complexity of $\mathcal{O}(\sqrt{kN})$. Our idea is to search a good threshold (the first phase) and use it for *finding all marked k-indices* algorithm (the second phase). We begin with presenting the second phase. In the second phase, all k' indices, where $k' \geq k$, are found. Suppose that threshold index $t_{k'}$ satisfy

$$M = \{x \mid g(x) < g(t_{k'}), x \in D\}, \tag{7}$$

$$|M| = k'. \tag{8}$$

In the first phase, in order to find the threshold $t_{k'}$, we use FM algorithm and QC algorithm. As shown in Sec. 2.2, in the process of FM algorithm, $\mathcal{O}(\sqrt{N})$ threshold indices are found in the descending order of values. Therefore, it is easy to find $t_{k'}$ from them by a binary search with QC algorithm.

We present more detail about this binary search with QC algorithm. Let us define the following.

- Let t_i^{FM} be the threshold index that is found in the i-th step of FM algorithm.
- Let T_{FM} be a set of thresholds that are found in the process of FM algorithm, which is given by

$$T_{FM} = \{t_1^{FM}, t_2^{FM}, \ldots\}$$

- Let M_i^{FM} be a set of marked indices of the i-th step in FM algorithm.
- Let $h(t)$ be a function that maps index t to the number of indices whose values are less than the value of threshold t in D, which is counted by QC algorithm. That is,

$$h(t) = |M(t)|, \tag{9}$$

where

$$M(t) = \{x \mid g(x) < g(t), x \in D\}. \tag{10}$$

The goal of the binary search is to find i such that $h(t_{i+1}^{FM}) \leq k < h(t_i^{FM})$. Once such i is found, $h(t_i^{FM})$ is used as k'. Fortunately, $t_{k'}$ exists in the last k indices of all found thresholds. Hence, we do not have to search all found $\mathcal{O}(\sqrt{N})$ indices but the last k indices. As QC algorithm requires $\mathcal{O}(\sqrt{N})$ query complexity for N indices [8], a binary search for k indices requires $\mathcal{O}(\log k)$ comparison. As QC algorithm has to run in each step of the binary search, threshold $t_{k'}$ can be found in $\mathcal{O}(\sqrt{N}\log k)$.

In summary, our threshold searching algorithm is shown below.

1. Apply FM algorithm and save the indices of the lastly found k thresholds.
2. Apply the binary search on the k indices of thresholds.

Once we find such a threshold, we can find all marked k' indices by applying *searching all marked k-indices* algorithm. This algorithm searches all elements in the set $\{x \mid x \in D, f(x) = 1\}$. For simplicity, we assume that $\mathcal{O}(k') = \mathcal{O}(k)$. Let T be a set of already found indices in the step of *searching all marked k indices algorithm* and let $f'_{t_k}(x)$ be an oracle such that

$$f'_{t_k}(x) = \begin{cases} 1, & \text{if } g(x) < g(t_k) \text{ and } x \notin T, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Then, searching all marked k-indices in M can be done in $\mathcal{O}(\sqrt{kN})$.

The whole algorithm of the proposed method is below.

1. Apply FM algorithm to D and save the last k indices of FM algorithm step.
2. Search threshold index $t_{k'}$ by a binary search on k indices. The comparison key is $|M^{FM}|$ that is derived by QC algorithm.
3. Apply *finding all marked k-indices* algorithm with threshold $t_{k'}$.

The total query complexity is given as

$$\mathcal{O}(\sqrt{N}\log k) + \mathcal{O}(\sqrt{kN}) = \mathcal{O}(\sqrt{kN}). \tag{12}$$

5 Conclusion

In this paper, we proposed a new *finding k-minima* algorithm and derived its query complexity. Our algorithm is easier to understand than Dürr's algorithm and written in a clear form. Our algorithm consists of two phases while many of others have one phase. Finding k-minima algorithm can be applied to many kinds of algorithms or applications. For example, k-nearest neighbor search, k-nearest neighbor clustering and classification. If these methods solve problems faster than the classical computer, the data that can be analyzed is greatly increased.

References

[1] Nathan Wiebe, Ashish Kapoor, and Krysta Svore. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *arXiv preprint arXiv:1401.2142*, 2014.

[2] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Quantum clustering algorithms. In *Proceedings of the 24th international conference on machine learning*, pages 1–8, 2007.

[3] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Quantum speed-up for unsupervised learning. *Machine Learning*, 90(2):261–287, 2013.

[4] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

[5] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[6] Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006.

[7] Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.

[8] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum counting. In *International Colloquium on Automata, Languages, and Programming*, pages 820–831. Springer, 1998.

[9] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In Jr. Samuel J. Lomonaco, editor, *Quantum Computation and Quantum Information*, volume 305, pages 53–74. American Mathematical Society, 2002.

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

[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, 1996.

[12] Andris Ambainis. Quantum search algorithms. *ACM SIGACT News*, 35(2):22–35, 2004.

[13] Andris Ambainis. A new quantum lower bound method, with an application to strong direct product theorem for quantum search. *arXiv preprint quant-ph/0508200*, 2005.

[14] Hartmut Klauck, Robert Špalek, and Ronald De Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM Journal on Computing*, 36(5):1472–1493, 2007.

[15] Sebastian Dörn and Thomas Thierauf. A note on the search for k elements via quantum walk. *Information Processing Letters*, 110(22):975–978, 2010.

[16] Ashish Ahuja and Sanjiv Kapoor. A quantum algorithm for finding the maximum. *arXiv preprint quant-ph/9911082*, 1999.