Quantum *k*-medoids algorithm using parallel amplitude estimation

Yong-Mei Li,[1] Hai-Ling Liu,[1] Shi-Jie Pan,[1] Su-Juan Qin,[1,*] Fei Gao,[1,†] Dong-Xu Sun,[2] and Qiao-Yan Wen[1]
[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*CNPC Digital and Information Management Department, Beijing 100007, China*

(Received 26 October 2022; revised 31 January 2023; accepted 2 February 2023; published 14 February 2023)

Quantum computing is a promising paradigm that can provide viable solutions to high-complexity problems. The *k*-medoids algorithm is a powerful clustering method ubiquitously used in data mining, image processing, pattern recognition, etc. The core of *k*-medoids algorithm is to perform cluster assignment and center update, which are time-consuming for large data sets. Aïmeur *et al.* proposed a quantum *k*-medoids algorithm [Aïmeur, Brassard, and Gambs, Mach. Learn. **90**, 261 (2013)] by quantizing the center update. Nevertheless, it has a query complexity $O(N^{3/2})$ for one iteration, which is computationally expensive for a large N where N is the number of points. In this paper, we propose a complete quantum algorithm for *k*-medoids algorithm. Specifically, in cluster assignment, we devise a quantum subroutine to calculate the Manhattan distance between any two points and then assign all points to the closest center in parallel, which is faster than what is achievable classically. In center update, for a cluster, we use parallel amplitude estimation to calculate the average distance of each point to all the others. It makes our algorithm polynomially faster than the algorithm of Aïmeur *et al.*, whose sum of distances of each point to all the others is computed by adding the distances one by one. Our quantum *k*-medoids algorithm, with time complexity $\widetilde{O}(N^{1/2})$, achieves a polynomial speedup in N compared to the existing one.

DOI: 10.1103/PhysRevA.107.022421

I. INTRODUCTION

Quantum computing can utilize quantum resources to process massive amounts of data both efficiently and securely, which has broad application prospects in information and computing. Tremendous progress is continuously being made both technologically and theoretically in it. Technologically, quantum hardware is making considerable advances [1–4]. Theoretically, considerable quantum algorithmic work is underway, such as cryptanalysis [5,6], to reduce the resources needed for implementing important classical algorithms. In recent years, a series of quantum algorithms were designed for machine learning problems in attempting to achieve potential quantum advantages, such as classification [7–9], clustering [10–15], linear regression [16–18], dimensionality reduction [19–23], matrix computation [24–26], and anomaly detection [27]. More progress on quantum machine learning algorithms can be found in Refs. [28,29].

Clustering is one of the most crucial unsupervised learning tasks, which refers to separating observed data into groups (i.e., clusters) with some quantified measurements, such that objects within a cluster are similar to each other but are dissimilar to objects in other clusters [30]. One of the most popular clustering algorithms is *k*-means [31] which iteratively finds the *k* centroids and assigns every object to the nearest centroid. The coordinate of each centroid is the mean of the coordinates of the objects in the cluster. Nevertheless, *k*-means is known to be very sensitive to outliers and noises. To avoid this problem, the *k*-medoids algorithm [32] is commonly used, where representative objects (or cluster centers) called medoids are considered instead of centroids. The medoid of a cluster is defined as the object within the cluster whose average (or sum) dissimilarity to all the other objects in this cluster is minimal. Similar to *k*-means, the core of *k*-medoids is to perform cluster assignment and center update. The *k*-medoids algorithm is widely used in such domains as data mining, image processing, and pattern recognition. But it works inefficiently for large data sets since its time complexity is $O(N^2Mk)$ for one iteration, where N is the number of points in the data set, M is the dimension of points, and k is the desired number of clusters. Therefore, it would be of great interest to design a quantum algorithm to reduce the complexity of the classical *k*-medoids algorithm.

A. Result

We develop a quantum *k*-medoids algorithm based on the Manhattan distance, in which we design two quantum subalgorithms to perform cluster assignment and center update, respectively. In cluster assignment, we devise a quantum subroutine to compute the Manhattan distance between any pair of points by quantum arithmetic operation [34–36] and then assign every point to the nearest center by a circuit for finding the minimum [33]. In center update, for a cluster, its new cluster center can be found by computing for each point inside the cluster its average (or sum) distance to all the other points and taking the minimum. Here we use quantum techniques, such as fixed-point quantum search [37] and parallel amplitude estimation [38,39], to calculate the average distance of each point to all the other points within the same cluster. Our quantum algorithm can be summarized as the following theorem.

*qsujuan@bupt.edu.cn
†gaof@bupt.edu.cn

TABLE I. The comparisons between our algorithm and the classical and quantum versions of the k-means/k-medoids algorithm. Here X is the data matrix, ϵ is an error parameter, x_{il} denotes the (i, l) entry of X, $\eta = \max_i(\|\mathbf{x}_i\|^2)$, $\mathbf{x}_i$ is the ith row of X, δ is a precision parameter, κ is the condition number of X, $\mu = \min_{p\in P}[\|X\|_F, \sqrt{s_{2p}(X)s_{2(1-p)}(X^T)}]$, where $P \subset [0, 1]$ such that $|P| = O(1)$ and $s_p(X) := \max_{i\in[N]} \|\mathbf{x}_i\|_p^p$.

Algorithm	Input	Output	Complexity[d]
Classical k-means [31]	$k, X \in R^{N\times M}$	k clusters and their centers	$O(kNM)$
Quantum k-means [12]	–	a quantum state corresponding to k clusters	$O[k\log(kNM)]$
Quantum k-means [33]	k, X stored in a QRAM, other parameters[a]	k centers	$\widetilde{O}[kM\frac{\eta}{\delta^2}\kappa(\mu + k\frac{\eta}{\delta}) + k^2\frac{\eta^{1.5}}{\delta^2}k\mu]$
Classical k-medoids [32]	k, X	k clusters and their centers	$O(kN^2M)$
Quantum k-medoids [10,11]	k, a distance oracle[b]	k clusters and their centers	$O(\frac{N^{3/2}}{\sqrt{k}})$
Our quantum k-medoids	k, X stored in a QRAM, ϵ	k centers and a quantum state corresponding to k clusters	$\widetilde{O}(\frac{k^{5/2}M^2N^{1/2}\max_{i,l}\|x_{il}\|}{\epsilon})$
		k clusters and their centers[c]	$\widetilde{O}(kNM + \frac{k^{5/2}M^2N^{1/2}\max_{i,l}\|x_{il}\|}{\epsilon})$

[a]The other parameters here denote the new parameters introduced by the quantum k-means algorithm in Ref. [33].
[b]See Sec. II B for more details.
[c]We can get the classical k clusters by performing a classical cluster assignment after obtaining k centers.
[d]We use time complexity to measure algorithm performance, except for the quantum k-medoids algorithm in Refs. [10,11], which use query complexity. For simplicity, we only consider the complexity of one iteration and use $\log(\cdot)$ to denote $\log_2(\cdot)$ throughout the paper. Note that with $\widetilde{O}$ we hide polylogarithmic factors.

Theorem 1. Given the data matrix $X \in R^{N\times M}$ stored in a quantum random access memory (QRAM) [40] and the parameter $\epsilon, k > 0$, the quantum k-medoids algorithm with high probability outputs k cluster centers and a quantum state corresponding to the k clusters in time $\widetilde{O}(\frac{k^{5/2}M^2N^{1/2}\max_{i,l}|x_{il}|}{\epsilon})$ per iteration, where ϵ is the error parameter for average distance estimation in center update and x_{il} denotes the (i, l) entry of X.

In conclusion, when $k, \max_{i,l}|x_{il}| = O(1)$, $M = \log_2 N$ and let $\frac{1}{\epsilon} = O[\log_2(NM)]$, the time complexity of our quantum k-medoids algorithm is $\widetilde{O}(N^{1/2})$ for one iteration, which achieves a polynomial speedup in N compared to the existing one [10,11], whose query complexity is $O(N^{3/2})$ under the same conditions. Note that our quantum algorithm can also be generalized to perform k-medoids clustering based on other distance measures such as Hamming distance and Chebyshev distance.

B. Related work

There is some work in quantum computing involving clustering problems. The quantum k-means algorithms in Refs. [12,33] achieve an exponential speedup over the classical k-means. The former belongs to the adiabatic quantum computing [41] and the latter utilizes the QRAM. The work most similar to ours is the quantum k-medoids algorithm proposed by Aïmeur *et al.* in Refs. [10,11], we call it the ABG algorithm. Based on the black-box model [42], the ABG algorithm uses a classical computer to perform cluster assignment and then quantum techniques to perform center update. It outputs the k clusters and their cluster centers. In our work, a trade-off is made between the amount of classical information obtained and the speed of the algorithm. Our quantum k-medoids algorithm outputs the k clusters centers and a quantum state corresponding to the k clusters. Unlike the ABG algorithm, our algorithm is an entire quantum algorithm, in which we redesign two quantum subalgorithms to perform cluster assignment and center update, respectively. In center update, the use of parallel amplitude estimation makes our algorithm polynomially faster than the ABG algorithm, whose sum of distances of each point to all the others is computed by adding the distances one by one. Of course, we can also obtain the classical information of k clusters by adding a classical cluster assignment. No matter what information we want to get, our quantum k-medoids algorithm will be faster than the existing one. See Table I for the comparisons between our algorithm and the classical and quantum versions of the k-means/k-medoids algorithm in an end-to-end setting.

The remainder of the paper is organized as follows. In Sec. II, we review the classical k-medoids algorithm and the ABG algorithm in Secs. II A and II B, respectively. In Sec. III A, we propose our quantum k-medoids algorithm in Sec. III A and analyze its time complexity in Sec. III B. Numerical simulations are reported in Sec. IV to validate the performance of our algorithm in practice. The conclusion is given in Sec. V.

II. REVIEW OF THE CLASSICAL k-MEDOIDS ALGORITHM AND THE ABG ALGORITHM

In this section, we will briefly review the classical k-medoids algorithm in Sec. II A, and the ABG algorithm in Sec. II B.

A. Review of the classical k-medoids algorithm

Let $X \in R^{N\times M}$ be a data set of points $\mathbf{x}_i \in R^M$ for $i \in [N]$, where $[N]$ denotes an index set $\{1, 2, \ldots, N\}$. For a predetermined parameter k, k-medoids clustering [32] aims to partition these points into k clusters according to a similarity measure, e.g., the Manhattan distance $d(\mathbf{x}_i, \mathbf{x}_s) = \|\mathbf{x}_i - \mathbf{x}_s\|_1$, where $\|\cdot\|_1$ is the ℓ_1 norm of a vector. See Fig. 1 for an

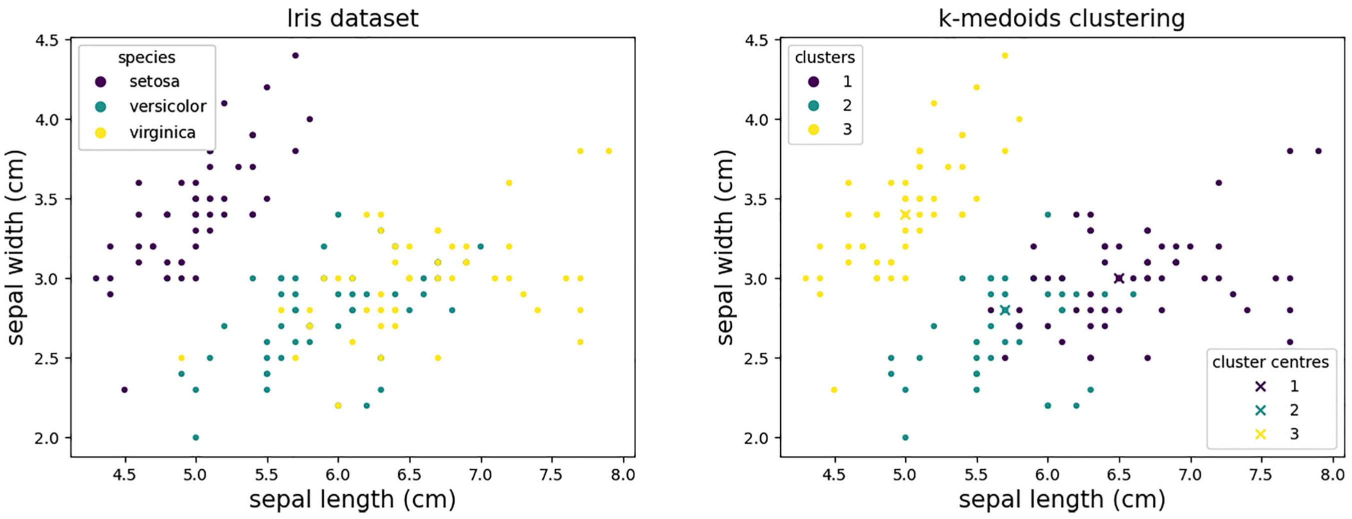

FIG. 1. The original Iris data set (left) and the result by k-medoids clustering (right).

example of clustering the well-known Iris data set. In general, the k-medoids problem is NP-hard to solve exactly for all $k \geqslant 2$ [43]. As such, many heuristic solutions exist. Here, we consider a Voronoi-iteration [31] k-medoids algorithm. The algorithm starts by selecting k initial centers at random among all points and then alternates between cluster assignment and center update until convergence. The convergence condition is that the minimum cost function is reached or the cluster centers are stabilized (or quasi-stabilized) or the maximum number of iterations is reached. Moreover, the k initial centers can also be selected by an initialization subroutine [44]. At iteration t, we denote the k clusters by the index sets $\mathcal{C}_j^t$ for $j \in [k]$, and the cluster center of $\mathcal{C}_j^t$ by the point $\mathbf{x}_{c_j^t}$. The process of classical k-medoids algorithm is shown in Algorithm 1.

The time complexity of the classical k-medoids algorithm is $O(N^2Mk)$ for one iteration, where N is the number of points in the data set, M is the dimension of points, and k is the desired number of clusters. It is computationally expensive when dealing with a large number of points. The ABG algorithm in the following subsection gives a feasible quantum acceleration scheme.

Algorithm 1. Classical k-medoids algorithm.

```
Input: Data matrix $X$, cluster number $k$.
Output: The $k$ clusters and their cluster centers.
  Step 1. Initialization
  Select $k$ initial centers at random among all points (or by
  an initialization subroutine).
  $t = 0$.
  repeat
    Step 2. Cluster assignment
    for each $i \in [N]$
      Compute the distances between point $\mathbf{x}_i$ and $k$
  cluster centers, and then attach $\mathbf{x}_i$ to its closest center.
    end for
    Step 3. Center update
    for each $j \in [k]$ do
      Find the medoid of the cluster $\mathcal{C}_j^t$ and make it its
  new center.
    end for
    $t = t + 1$.
until convergence condition is satisfied.
return The $k$ clusters and their cluster centers.
```

B. Review of the ABG algorithm

In Refs. [10,11], the authors assume that the distance (or a similarity measure) between points of the data set is available solely through a black box (also called an oracle) as shown in Fig. 2. Using this oracle, they build the quantum circuit illustrated in Fig. 3, which takes $|i\rangle$ as the input, $1 \leqslant i \leqslant m$, and computes the sum of the distances between $\mathbf{x}_i$ and all the other points within the cluster $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$. The quantum minimum-finding algorithm [45] can then be used to find the minimum such sum over all possible $\mathbf{x}_i$. It is possible to compute the medoid of a cluster by using the quantum subroutine as described above. Based on this, they use a classical computer to perform cluster assignment and then quantum techniques to find the new cluster centers. Finally, the ABG algorithm outputs the k clusters and their centers.

For simplicity, they assume that the clusters have roughly the same size $\frac{N}{k}$. This yields a query complexity $O(\frac{N^{3/2}}{\sqrt{k}})$ for one iteration [10,11]. The complexity of the ABG algorithm depends on $N^{3/2}$, which makes it unsuitable for large data sets. In the following section, we will introduce a new quantum k-medoids algorithm that can significantly reduce the complexity of the ABG algorithm.

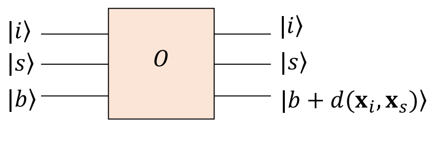

FIG. 2. Illustration of the distance oracle. The addition $b + d(\mathbf{x}_i, \mathbf{x}_s)$ is performed in an appropriate finite group between the ancillary register $|b\rangle$ and the distance $d(\mathbf{x}_i, \mathbf{x}_s)$.

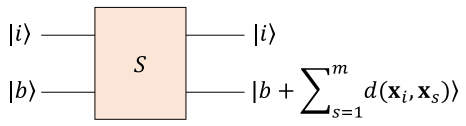

FIG. 3. Computing the sum of distances between $\mathbf{x}_i$ and all the other points within the cluster $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$. The oracle S can be obtained by repeating m times the oracle O described in Fig. 2 for s ranging from 1 to m.

III. QUANTUM *k*-MEDOIDS ALGORITHM

In this section, we present a quantum k-medoids algorithm in Sec. III A and analyze its complexity in Sec. III B.

Our quantum k-medoids algorithm follows the same steps as the classical k-medoids algorithm. In step 1, we pick k initial centers at random among all points and then store their indexes in a QRAM. In step 2, we compute the Manhattan distances between all points and the k cluster centers, then all points are assigned to the closest center in superposition. In step 3, for each cluster, we find the point within the cluster whose average distance to all the other points in this cluster is minimum, then we update the above QRAM with the indexes of the k points we found, repeating the last two steps until convergence. An overview of our algorithm is shown as Algorithm 2.

A. Algorithm

Assume that the data matrix $X \in R^{N\times M}$ is stored in a QRAM [40] which allows the following mapping to be performed in time $O[\log(NM)]$:

$$\mathbf{O}_X : |i\rangle|l\rangle|0\rangle \rightarrow |i\rangle|l\rangle|x_{il}\rangle, \quad (1)$$

where x_{il} denotes the (i, l) entry of X.

In addition, at iteration t, the index vector $\mathbf{c}^t := [c_1^t, c_2^t, \ldots, c_k^t]^T$ is stored in a QRAM, that is, the following mapping can be performed in time $O(\log k)$:

$$\mathbf{O}_c^t : |j\rangle|0\rangle \rightarrow |j\rangle|c_j^t\rangle, \quad (2)$$

where c_j^t is the index of the center of $\mathcal{C}_j^t$.

For ease of understanding, here we introduce two lemmas which are necessary for our quantum algorithm.

Lemma 1. (Manhattan distance calculation). Given a unitary $\mathbf{O}_X : |i\rangle|l\rangle|0\rangle \rightarrow |i\rangle|l\rangle|x_{il}\rangle$ which can be performed in time $O[\log(NM)]$. Then, there exists a quantum algorithm that performs the following mapping:

$$Q_1 : |i\rangle|s\rangle|0\rangle \rightarrow |i\rangle|s\rangle|d(\mathbf{x}_i, \mathbf{x}_s)\rangle \quad (3)$$

in time $O[M\log(NM)]$, where $d(\mathbf{x}_i, \mathbf{x}_s)$ is the Manhattan distance between two points $\mathbf{x}_i$ and $\mathbf{x}_s$.

Proof. See Appendix A. ■

Lemma 2. (Circuit for finding the minimum [33]). Given k different $\log q$-bit registers $\bigotimes_{j=1}^k |d_j\rangle$, there is a quantum circuit that maps

$$\left(\bigotimes_{j=1}^{k} |d_j\rangle\right)|0\rangle \rightarrow \left(\bigotimes_{j=1}^{k} |d_j\rangle\right)\left|\arg\min_j(d_j)\right\rangle \quad (4)$$

in time $O(k\log q)$.

Algorithm 2. Quantum k-medoids algorithm.

```
Input: Data matrix $X$ stored in a QRAM. Cluster number
  $k$, error parameter $\epsilon$ for average distance estimation.
Output: The $k$ clusters centers and a quantum state corresponding
  to the $k$ clusters.
  Step 1. Initialization
  Select $k$ initial centers at random among all points and store
  the initial index vector $\mathbf{c}^0 = [c_1^0, c_2^0, \ldots, c_k^0]^T$ in a QRAM.
  $t = 0$.
  repeat
    Step 2. Cluster assignment
    (2.1) Prepare the state $\sum_{i=1}^N \frac{1}{\sqrt{N}}|i\rangle\bigotimes_{j=1}^k(|j\rangle|c_j^t\rangle|0\rangle^{\otimes\lceil\log q\rceil})$;
    (2.2) Compute the distances between all points and the
    $k$ centers to get $\frac{1}{\sqrt{N}}\sum_{i=1}^N |i\rangle\bigotimes_{j=1}^k[|j\rangle|d(\mathbf{x}_i, \mathbf{x}_{c_j^t})\rangle]$;
    (2.3) Find the minimum among $\{d(\mathbf{x}_i, \mathbf{x}_{c_j^t})\}_{j\in[k]}$ to create
    the superposition of all points and their cluster labels:
    $|\phi^t\rangle = \frac{1}{\sqrt{N}}\sum_{i=1}^N |i\rangle|j^t(\mathbf{x}_i)\rangle$.
    Step 3. Center update
    for each $j \in [k]$ do
      (3.1) Perform the fixed-point quantum search
      algorithm on the state $|\phi^t\rangle|0\rangle|\phi^t\rangle$ to prepare the state
      $\frac{1}{\sqrt{|\mathcal{C}_j^t|}}\sum_{i\in\mathcal{C}_j^t}|i\rangle|0\rangle\frac{1}{\sqrt{|\mathcal{C}_j^t|}}\sum_{s\in\mathcal{C}_j^t}|s\rangle$;
      (3.2) For a given error $\epsilon$, estimate the average
      distance of $\mathbf{x}_i$ to all the other points within $\mathcal{C}_j^t$ to create the
      state $\frac{1}{\sqrt{|\mathcal{C}_j^t|}}\sum_{i\in\mathcal{C}_j^t}|i\rangle|\frac{\sum_{s\in\mathcal{C}_j^t} d(\mathbf{x}_i,\mathbf{x}_s)}{|\mathcal{C}_j^t|}\rangle$;
      (3.3) Find the minimum among $\{\frac{\sum_{s\in\mathcal{C}_j^t} d(\mathbf{x}_i,\mathbf{x}_s)}{|\mathcal{C}_j^t|}\}_{i\in\mathcal{C}_j^t}$
      and then let $c_j^{t+1} = \arg\min_{i\in\mathcal{C}_j^t}(\frac{\sum_{s\in\mathcal{C}_j^t} d(\mathbf{x}_i,\mathbf{x}_s)}{|\mathcal{C}_j^t|})$;
    end for
    (3.4) Update the QRAM for the index vector with the
    new vector $\mathbf{c}^{t+1} = [c_1^{t+1}, c_2^{t+1}, \ldots, c_k^{t+1}]^T$.
    $t = t + 1$.
until convergence condition is satisfied.
return The $k$ clusters centers and a quantum state corresponding
to the $k$ clusters.
```

The above lemma can be easily generalized to the following quantum circuit:

$$U_{\min} : \left(\bigotimes_{j=1}^{k} |j\rangle|d_j\rangle\right)|0\rangle \rightarrow \left(\bigotimes_{j=1}^{k} |j\rangle|d_j\rangle\right)\left|\arg\min_j(d_j)\right\rangle. \quad (5)$$

Now we detail the process of our quantum k-medoids algorithm.

Step 1. Initialization

Here, we select k initial centers at random among all points and then store the initial index vector $\mathbf{c}^0 = [c_1^0, c_2^0, \ldots, c_k^0]^T$ in a QRAM. Moreover, the initial cluster centers can be chosen by a quantum initialization algorithm in Ref. [11].

Step 2. Cluster assignment

At iteration t, our quantum subalgorithm for cluster assignment consists of the following three stages. Among them, we first carry out stages (2.1) and (2.2) to compute the Manhattan distances between all points and the k cluster centers, then implement stage (2.3) to assign all points to the closest center

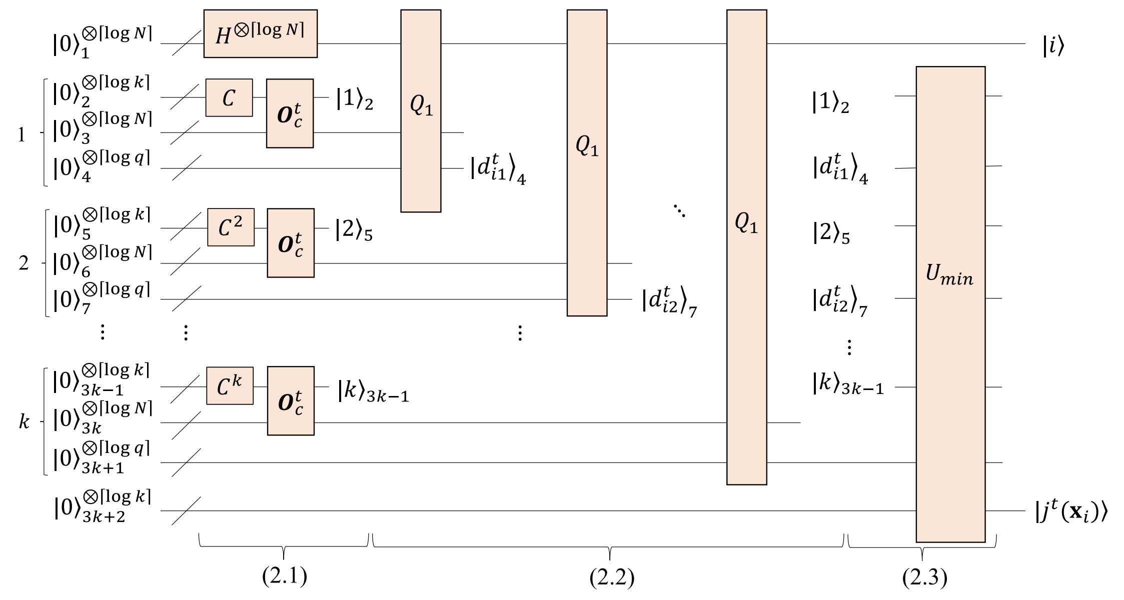
$|0\rangle_1^{\otimes\lceil\log N\rceil}$
$H^{\otimes\lceil\log N\rceil}$
$|0\rangle_2^{\otimes\lceil\log k\rceil}$
$|0\rangle_3^{\otimes\lceil\log N\rceil}$
$|0\rangle_4^{\otimes\lceil\log q\rceil}$
C
$\boldsymbol{O}_c^t$
$|1\rangle_2$
Q_1
$|d_{i1}^t\rangle_4$
$|0\rangle_5^{\otimes\lceil\log k\rceil}$
$|0\rangle_6^{\otimes\lceil\log N\rceil}$
$|0\rangle_7^{\otimes\lceil\log q\rceil}$
C^2
$|2\rangle_5$
$|d_{i2}^t\rangle_7$
$|0\rangle_{3k-1}^{\otimes\lceil\log k\rceil}$
$|0\rangle_{3k}^{\otimes\lceil\log N\rceil}$
$|0\rangle_{3k+1}^{\otimes\lceil\log q\rceil}$
C^k
$|k\rangle_{3k-1}$
$|0\rangle_{3k+2}^{\otimes\lceil\log k\rceil}$
U_{min}
$|i\rangle$
$|j^t(\mathbf{x}_i)\rangle$
1
2
k
(2.1)
(2.2)
(2.3)

FIG. 4. Quantum circuit of step 2 of our algorithm. Here the numbers $1, 2, \ldots, k$ at the left-most denote the sequence of k sets of registers in the tensor product state $\bigotimes_{j=1}^{k}(|j\rangle|0\rangle^{\otimes\lceil\log N\rceil}|0\rangle^{\otimes\lceil\log q\rceil})$, (2.1), (2.2), and (2.3) denote the three stages of step 2, "/" denotes a bundle of wires, H denotes the Hadamard gate, and d_{ij}^t denotes $d(\mathbf{x}_i, \mathbf{x}_{c_j^t})$.

in superposition. At the end of cluster assignment, we obtain the superposition of all points and their cluster labels. The details are as follows.

(2.1) Prepare the state

$$\sum_{i=1}^{N}\frac{1}{\sqrt{N}}|i\rangle\bigotimes_{j=1}^{k}\left(|j\rangle|c_j^t\rangle|0\rangle^{\otimes\lceil\log q\rceil}\right), \tag{6}$$

where $q = 2M\max_{i,l}|x_{il}|$, $\lceil\cdot\rceil$ is the ceiling function, the tensor product state $\bigotimes_{j=1}^{k}(|j\rangle|c_j^t\rangle|0\rangle^{\otimes\lceil\log q\rceil})$ corresponds to the k cluster centers and will be used to select the nearest center for each point.

To get the above state, we first prepare the initial state

$$\sum_{i=1}^{N}\frac{1}{\sqrt{N}}|i\rangle\bigotimes_{j=1}^{k}(|0\rangle^{\otimes\lceil\log k\rceil}|0\rangle^{\otimes\lceil\log N\rceil}|0\rangle^{\otimes\lceil\log q\rceil}). \tag{7}$$

Then, we perform the unitary $I^{\otimes\lceil\log N\rceil}\bigotimes_{j=1}^{k}(C^j\otimes I^{\otimes(\lceil\log N\rceil+\lceil\log q\rceil)})$ on the above state to get

$$\sum_{i=1}^{N}\frac{1}{\sqrt{N}}|i\rangle\bigotimes_{j=1}^{k}(|j\rangle|0\rangle^{\otimes\lceil\log N\rceil}|0\rangle^{\otimes\lceil\log q\rceil}), \tag{8}$$

where C is a circular shift operator that performs the mapping $C: |j-1\rangle \to |j\rangle$ for $j \in [k]$. After that, the target state can be obtained by calling $\mathbf{O}_c^t$.

(2.2) Compute the Manhattan distances between all points and the k cluster centers by Q_1 (lemma 1), and then discard all $|c_j^t\rangle$ to get the state $\frac{1}{\sqrt{N}}\sum_{i=1}^{N}|i\rangle\bigotimes_{j=1}^{k}[|j\rangle|d(\mathbf{x}_i,\mathbf{x}_{c_j^t})\rangle]$.

(2.3) Invoke $U_{\min}$ to find the minimum distance among $\{d(\mathbf{x}_i,\mathbf{x}_{c_j^t})\}_{j\in[k]}$ and then uncompute the redundant registers to create the superposition of all points and their cluster labels:

$$|\phi^t\rangle := \frac{1}{\sqrt{N}}\sum_{i=1}^{N}|i\rangle|j^t(\mathbf{x}_i)\rangle, \tag{9}$$

where $j^t(\mathbf{x}_i) = \arg\min_{j\in[k]}[d(\mathbf{x}_i,\mathbf{x}_{c_j^t})]$ is the cluster label of point $\mathbf{x}_i$ at iteration t.

The entire quantum circuit of step 2 is shown in Fig. 4.

Step 3. Center update

At iteration t, our quantum subalgorithm for center update consists of the following four stages. Among them, we first carry out stages (3.1)–(3.3) for each $j \in [k]$ to find the k medoids, then implement (3.4) to update the QRAM for the index vector $\mathbf{c}^t$. The details are as follows.

(3.1) Prepare the state

$$\frac{1}{\sqrt{|\mathcal{C}_j^t|}}\sum_{i\in\mathcal{C}_j^t}|i\rangle|0\rangle\frac{1}{\sqrt{|\mathcal{C}_j^t|}}\sum_{s\in\mathcal{C}_j^t}|s\rangle, \tag{10}$$

where $|\mathcal{C}_j^t|$ denotes the number of points in cluster $\mathcal{C}_j^t$.

Note that, at the end of stage (2.3) in step 2, we get the state $|\phi^t\rangle$, which can be rewritten as

$$|\phi^t\rangle = \sum_{j=1}^{k}\sqrt{\frac{|\mathcal{C}_j^t|}{N}}\left(\frac{1}{\sqrt{|\mathcal{C}_j^t|}}\sum_{i\in\mathcal{C}_j^t}|i\rangle\right)|j\rangle. \tag{11}$$

Based on this, we can first prepare the state $|\phi^t\rangle|0\rangle|\phi^t\rangle$. For the target state having the same state, i.e., $|j\rangle$, as appears in the last register of both $|\phi^t\rangle$, we apply the fixed-point quantum search algorithm proposed in Ref. [37] to amplify the amplitude of it. Ideally, we get the following state:

$$\frac{1}{\sqrt{|\mathcal{C}_j^t|}}\sum_{i\in\mathcal{C}_j^t}|i\rangle|j\rangle|0\rangle\frac{1}{\sqrt{|\mathcal{C}_j^t|}}\sum_{s\in\mathcal{C}_j^t}|s\rangle|j\rangle. \tag{12}$$

Also, in Appendix B, we discuss the case that the above quantum state is obtained with a certain successful probability. After that, we obtain the target state by discarding the second and fifth registers.

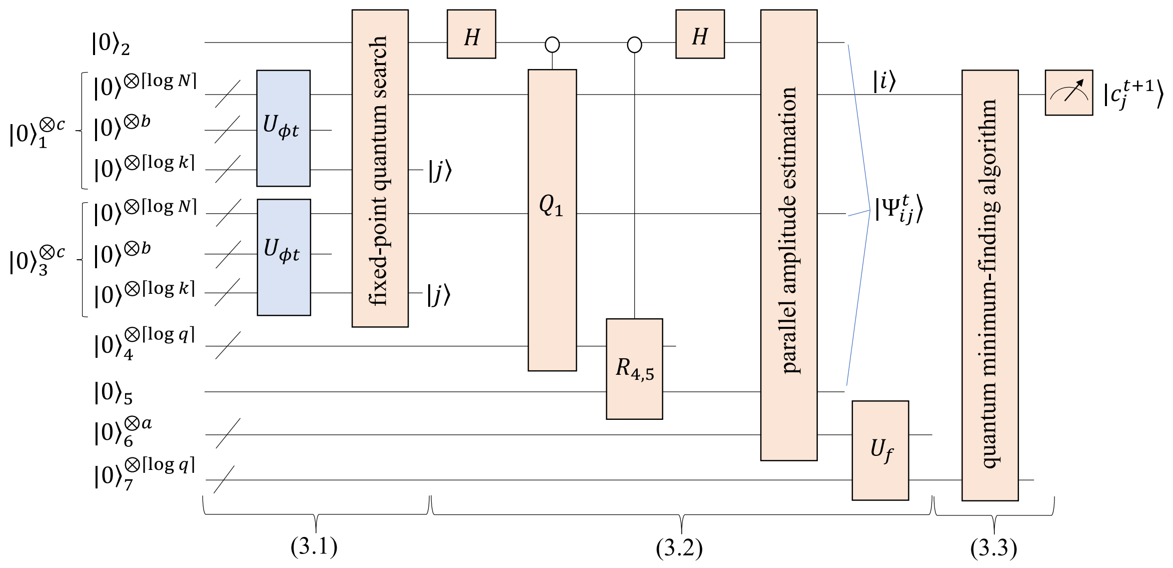
$|0\rangle_2$
$|0\rangle_1^{\otimes c}$
$|0\rangle^{\otimes\lceil\log N\rceil}$
$|0\rangle^{\otimes b}$
$|0\rangle^{\otimes\lceil\log k\rceil}$
$|0\rangle_3^{\otimes c}$
$|0\rangle_4^{\otimes\lceil\log q\rceil}$
$|0\rangle_5$
$|0\rangle_6^{\otimes a}$
$|0\rangle_7^{\otimes\lceil\log q\rceil}$
$U_{\phi t}$
fixed-point quantum search
$|j\rangle$
H
Q_1
$R_{4,5}$
parallel amplitude estimation
U_f
$|i\rangle$
$|\Psi_{ij}^t\rangle$
quantum minimum-finding algorithm
$|c_j^{t+1}\rangle$
(3.1)
(3.2)
(3.3)

FIG. 5. Quantum circuit of stages (3.1)–(3.3) in step 3 of our algorithm. Here, $U_{\phi t}$ is the unitary operation for preparing the quantum state $|\phi^t\rangle$ and its quantum circuit is shown in Fig. 4, $c = b + \lceil\log N\rceil + \lceil\log k\rceil$ and $b = k(\lceil\log k\rceil + \lceil\log N\rceil + \lceil\log q\rceil)$.

(3.2) Estimate the average distance of $\mathbf{x}_i$ to all the other points in $\mathcal{C}_j^t$ with error ϵ, then uncompute the redundant registers to create the state

$$\frac{1}{\sqrt{|\mathcal{C}_j^t|}}\sum_{i\in\mathcal{C}_j^t}|i\rangle\left|\frac{\sum_{s\in\mathcal{C}_j^t}d(\mathbf{x}_i,\mathbf{x}_s)}{|\mathcal{C}_j^t|}\right\rangle. \tag{13}$$

The core of stage (3.2) is a fast quantum method for computing the average distance via the inner product, which we couple with parallel amplitude estimation. We first prepare a particular quantum state as shown in Eq. (C6), where the distance information is stored as amplitudes of it. After that, we perform parallel amplitude estimation on this quantum state to get the value of the inner product $\frac{\sum_{s\in\mathcal{C}_j^t}d(\mathbf{x}_i,\mathbf{x}_s)}{q|\mathcal{C}_j^t|}$, and then get the average distance $\frac{\sum_{s\in\mathcal{C}_j^t}d(\mathbf{x}_i,\mathbf{x}_s)}{|\mathcal{C}_j^t|}$ through a reversible circuit. The specific process is depicted in Appendix C.

(3.3) Invoke a quantum minimum-finding algorithm to find the minimum among the set $\{\frac{\sum_{s\in\mathcal{C}_j^t}d(\mathbf{x}_i,\mathbf{x}_s)}{|\mathcal{C}_j^t|}\}_{i\in\mathcal{C}_j^t}$ [13,45] and then let $c_j^{t+1} = \arg\min_{i\in\mathcal{C}_j^t}(\frac{\sum_{s\in\mathcal{C}_j^t}d(\mathbf{x}_i,\mathbf{x}_s)}{|\mathcal{C}_j^t|})$.

The entire quantum circuit of stages (3.1)–(3.3) is shown in Fig. 5.

(3.4) After performing stages (3.1)–(3.3) for each $j \in [k]$, we get $c_1^{t+1}, c_2^{t+1}, \ldots, c_k^{t+1}$ which are the indexes of the k new cluster centers (i.e., medoids). Then, we update the QRAM for the index vector, i.e., update the index vector with the vector $\mathbf{c}^{t+1} = [c_1^{t+1}, c_2^{t+1}, \ldots, c_k^{t+1}]^T$, and store it in the QRAM for the next iteration.

Step 2 and step 3 alternate until the convergence condition is satisfied. Once we obtain the stable k centers, we get the quantum state corresponding to all points and their cluster labels by using the quantum subalgorithm for cluster assignment. Finally, our quantum algorithm outputs the k centers and a quantum state corresponding to the k clusters.

Note that if we want to obtain the classical information of k clusters, we can perform a classical cluster assignment after obtaining the stable k centers instead of using the quantum subalgorithm.

B. Complexity analysis

The time complexity of our algorithm is mainly from steps 2 and 3. Now we respectively analyze their complexity and discuss the overall complexity.

In step 2, for stage (2.1), we use $\lceil\log N\rceil$ Hadamard gates to prepare the initial state and then implement $\frac{k(k+1)}{2}$ circular shift operators. The target state can be obtained by calling $\mathbf{O}_c^t$ for k times, with a run time of $O(\log k)$ for each call. The total complexity is $O(\lceil\log N\rceil + \frac{k(k+1)}{2} + k\log k)$. For stage (2.2), we should invoke Q_1 for k times to compute the distances between all points and the k cluster centers, hence its complexity is $O[kM\log(NM)]$ by lemma 1. By lemma 2, the cost of finding the minimum is $O(k\lceil\log q\rceil)$ in stage (2.3). In total, the time complexity of step 2 is $O[kM\log(NM)]$.

In step 3, similar to the ABG algorithm, we assume that all clusters have roughly size $\Theta(\frac{N}{k})$. For stage (3.1), with state $|\phi^t\rangle|0\rangle|\phi^t\rangle$, invoking $O(k)$ times the fixed-point quantum search is enough to obtain the target state. For stage (3.2), the time complexity of Hadamard gates and controlled rotation can be neglected compared with other subroutines. The time complexity of implementing the Grover operator G is mainly from the unitary U, which is equal to the complexity of stages (3.1)–(3.2)(iii). Hence, the parallel amplitude estimation has a query complexity of $O[\frac{1}{\epsilon_A}(2 + \frac{1}{2\eta})]$ and each query has a time complexity $O[k^2M\log(NM)]$, where ϵ_A is the error of amplitude estimation and $1-\eta$ is the probability to succeed. Suppose we wish to approximate $\frac{\theta_{ij1}^t}{\pi}$ or $1-\frac{\theta_{ij1}^t}{\pi}$ to an accuracy 2^{-n} with probability of success at least $1-\eta$, we should choose $a = n + \lceil\log(2 + \frac{1}{2\eta})\rceil$ [46]. After the parallel amplitude estimation, we obtain the value of $\frac{\theta_{ij1}^t}{\pi}$ or $1-\frac{\theta_{ij1}^t}{\pi}$ with error ϵ_{ij}^t, and then we can easily compute the average distance

TABLE II. The time complexity of each step of our algorithm in one iteration. Here we neglect the runtime of step 1 and stage (3.4).

Steps	Stages	Time complexity
Step 2	(2.1)	$O(\lceil\log N\rceil + \frac{k(k+1)}{2} + k\log k)$
	(2.2)	$O[kM\log(NM)]$
	(2.3)	$O(k\lceil\log q\rceil)$
	All stages	$O[kM\log(NM)]$
Step 3	(3.1)	$O[k^2M\log(NM)]$
	(3.2)	$O(\frac{k^2M^2\log(NM)\max_{i,l}\lvert x_{il}\rvert}{\epsilon})$
	(3.3)	$\widetilde{O}(\frac{k^{3/2}M^2N^{1/2}\max_{i,l}\lvert x_{il}\rvert}{\epsilon})$
	All stages	$\widetilde{O}(\frac{k^{5/2}M^2N^{1/2}\max_{i,l}\lvert x_{il}\rvert}{\epsilon})$

$\frac{\sum_{s\in C_j^t} d(\mathbf{x}_i,\mathbf{x}_s)}{|C_j^t|} = q(1-2\sin^2\theta_{ij1}^t)$ by U_f. And its error is

$$\begin{aligned}
&2q\left|\sin^2\left(\theta_{ij1}^t+\pi\epsilon_{ij}^t\right)-\sin^2\theta_{ij1}^t\right|\\
&\quad= 2q\left|\sin\left(\theta_{ij1}^t+\pi\epsilon_{ij}^t\right)+\sin\theta_{ij1}^t\right|\left|\sin\left(\theta_{ij1}^t+\pi\epsilon_{ij}^t\right)-\sin\theta_{ij1}^t\right|\\
&\quad\leqslant 2q\left|2\sin\theta_{ij1}^t+\pi\epsilon_{ij}^t\right|\left|\pi\epsilon_{ij}^t\right|\\
&\quad\leqslant 4q\pi\left|\epsilon_{ij}^t\right|+2q\left(\pi\epsilon_{ij}^t\right)^2,\\
&\quad\leqslant 4q\pi|\epsilon_A|+2q(\pi\epsilon_A)^2, \qquad (14)
\end{aligned}$$

where the first inequality holds by $\sin(\theta_{ij1}^t+\pi\epsilon_{ij}^t)\leqslant \sin\theta_{ij1}^t+\pi\epsilon_{ij}^t$. If we want to have the average distance in the end with an absolute error ϵ, we can control the error of parallel amplitude estimation as $\frac{\epsilon}{4q\pi}$, that is, $\epsilon_A = \frac{\epsilon}{4q\pi}$. Therefore, the total time complexity of stage (3.2) is $O(\frac{k^2M^2\log(NM)\max_{i,l}|x_{il}|}{\epsilon})$. For stage (3.3), given an oracle for preparing the state $\frac{1}{\sqrt{|C_j^t|}}\sum_{i\in C_j^t}|i\rangle|\frac{\sum_{s\in C_j^t} d(\mathbf{x}_i,\mathbf{x}_s)}{|C_j^t|}\rangle$ with the successful probability $1-\eta$, the expected number of queries made to find the minimum with failure probability at most δ is bounded above by roughly $90\sqrt{N/k}\lceil\frac{\log(\frac{81\sqrt{N/k}(\log\sqrt{N/k}+\gamma)}{\delta})}{2(\frac{1}{2}-\eta)^2}\rceil$, where $\gamma\approx 0.5772$ is Euler's constant. A detailed complexity analysis is provided in Ref. [13]. For simplicity, here we could simply choose $\delta,\eta = O(1)$, and the query complexity of (3.3) can then be reduced to $\widetilde{O}(\sqrt{N/k})$. Before stage (3.4), we should perform stages (3.1)–(3.3) for each $j\in[k]$, that is, stages (3.1)–(3.3) should be repeated for k times to get the k new cluster centers. The time complexity of (3.4) can be omitted compared to other stages. In total, the time complexity of step 3 is $\widetilde{O}(\frac{k^{5/2}M^2N^{1/2}\max_{i,l}|x_{il}|}{\epsilon})$.

The complexity of each step of our algorithm in one iteration is summarized as Table II.

As a conclusion, the overall time complexity of the our algorithm is $\widetilde{O}(\frac{k^{5/2}M^2N^{1/2}\max_{i,l}|x_{il}|}{\epsilon})$ for one iteration. If $\max_{i,l}|x_{il}| = O(1)$ and let $\frac{1}{\epsilon} = O[\log(NM)]$, it can be reduced to $\widetilde{O}(k^{5/2}M^2N^{1/2})$.

Assuming that there exists an oracle that can be used to query the distance between two points, the query complexity of ABG algorithm is $O(\frac{N^{3/2}}{\sqrt{k}})$ for one iteration [10,11]. The time complexity of our quantum algorithm is $\widetilde{O}(N^{1/2})$ for one iteration when $k = O(1)$ and $M = \log N$, which achieves a polynomial speedup in N over the ABG algorithm whose query complexity is $O(N^{3/2})$.

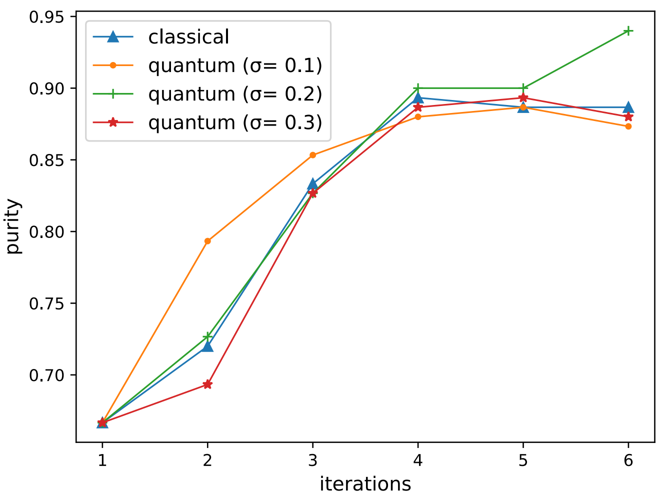

FIG. 6. Purity evolution on the Iris data set under classical k-medoids and quantum k-medoids (i.e., k-medoids with noise) with different noises. We added noise on the average distance, in which the noise is selected randomly from Gaussian noise with a mean of 0 and a standard deviation of σ.

Note that if we want to obtain the classical information of k clusters rather than the quantum information, an additional classical cluster assignment is needed. The time complexity of the classical cluster assignment is $O(kNM)$. In this way, our algorithm can obtain the classical information like in the ABG algorithm but in a shorter runtime.

IV. NUMERICAL SIMULATIONS

In this section, we would like to demonstrate that our quantum k-medoids algorithm provides good clustering results. Limited by the capabilities of existing quantum computers, these simulations are made with a classical computer. Our quantum k-medoids algorithm follows the same steps as the classical k-medoids algorithm, and only introduces the error ϵ in the stage of average distance estimation. The complexity analysis in Sec. III B provides theoretical evidence that the value of ϵ is related to the time complexity of our algorithm and we can run the algorithm long enough [roughly in time $\widetilde{O}(N^{\frac{1}{2}})$] to control it in an acceptable range. Thus, our quantum k-medoids algorithm can be viewed as a quantum equivalent of the classical k-medoids algorithm with noise. Based on this, we used a classical computer to simulate the quantum steps and introduced equivalent noise and randomness in average distance. We ran the k-medoids and the quantum k-medoids (i.e., k-medoids with noise) for different values of noise on the well-known Iris data set. Experimental results are shown in Fig. 6.

The Iris data set has three classes, i.e., setosa, virginica, and versicolor, of size 50 each (four dimensions each). In this numerical experiment, we used purity [47] to measure the performance of the clustering algorithm. The purity of clustering is similar to the accuracy of classification. All experiments started with the same initial centers. It follows from Fig. 6 that for different values of the noise, both k-medoids and quantum k-medoids reached a similar purity after the fourth iteration.

V. CONCLUSION

In conclusion, we have proposed the quantum k-medoids algorithm, which achieves a polynomial speedup in the number of points over the existing one under certain conditions.

Lemma 1 provided an efficient method to compute the Manhattan distance between any two points, which can be reused as a subroutine for other quantum algorithms. Moreover, it can also be modified to compute other distance measures such as the Euclidean distance, Hamming distance, and Chebyshev distance. Finally and most importantly, in step 3 of our algorithm, the reason we can calculate the average distance of a point to all the other points inside the cluster by the parallel amplitude estimation is that we have managed to encode the distance information into the amplitude of the computational basis states. The parallel amplitude estimation is a powerful tool for solving the problem whose solution can be encoded into the amplitude of a particular quantum state. This is the main idea of step 3 of our algorithm. We believe that this idea could also be applied to solve other machine learning problems, such as density estimation and data classification. We hope the techniques and ideas we used in this paper will inspire others in the field of quantum machine learning.

ACKNOWLEDGMENTS

This work is supported by the Beijing Natural Science Foundation (Grant No. 4222031) and the National Natural Science Foundation of China (Grants No. 61976024, No. 61972048, and No. 62171056).

APPENDIX A: PROOF OF LEMMA 1

Here we show how to implement the unitary Q_1 in lemma 1. Let us start with the initial state

$$|i\rangle|s\rangle|0\rangle^{\otimes\lceil\log q\rceil}\bigotimes_{l=1}^{M}\left(|0\rangle^{\otimes\lceil\log M\rceil}|0\rangle^{\otimes(\lceil\log q_1\rceil+1)}|0\rangle^{\otimes(\lceil\log q_1\rceil+2)}\right), \tag{A1}$$

where $q = 2Mq_1$ and $q_1 = \max_{i\in[N],l\in[M]}|x_{il}|$. Using a unitary $U_f : |x\rangle|0\rangle^{\otimes\lceil\log M\rceil} \to |x\rangle|f(x)\rangle$ for M times, where $f(x)$ can be calculated efficiently in classical, we can perform the mapping

$$|i\rangle|s\rangle|x\rangle\bigotimes_{l=1}^{M}\left(|0\rangle^{\otimes\lceil\log M\rceil}|0\rangle^{\otimes(\lceil\log q_1\rceil+1)}|0\rangle^{\otimes(\lceil\log q_1\rceil+2)}\right)$$
$$\to|i\rangle|s\rangle|x\rangle\bigotimes_{l=1}^{M}\left(|f(x)\rangle|0\rangle^{\otimes(\lceil\log q_1\rceil+1)}|0\rangle^{\otimes(\lceil\log q_1\rceil+2)}\right). \tag{A2}$$

Based on this, we get

$$|i\rangle|s\rangle|0\rangle^{\otimes\lceil\log q\rceil}\bigotimes_{l=1}^{M}\left(|l\rangle|0\rangle^{\otimes(\lceil\log q_1\rceil+1)}|0\rangle^{\otimes(\lceil\log q_1\rceil+2)}\right) \tag{A3}$$

by performing $U_f : |x\rangle|0\rangle^{\otimes\lceil\log M\rceil} \to |x\rangle|x+l\rangle$ on the initial state for each $l \in [M]$, where $x = 0$.

Then, we query the state preparation oracle $\mathbf{O}_X$ to get

$$|i\rangle|s\rangle|0\rangle^{\otimes\lceil\log q\rceil}\bigotimes_{l=1}^{M}(|l\rangle|x_{il}\rangle|x_{sl}\rangle). \tag{A4}$$

Next, we perform quantum arithmetic operation [34–36] on the above state to get

$$|i\rangle|s\rangle|0\rangle^{\otimes\lceil\log q\rceil}\bigotimes_{l=1}^{M}(|l\rangle|x_{il}\rangle|x_{il}-x_{sl}\rangle). \tag{A5}$$

After that, we perform a QFT-based absolute value operation [36] to yield

$$|i\rangle|s\rangle|0\rangle^{\otimes\lceil\log q\rceil}\bigotimes_{l=1}^{M}(|l\rangle|x_{il}\rangle||x_{il}-x_{sl}|\rangle). \tag{A6}$$

Finally, we add up $|x_{il}-x_{sl}|$ in each dimension by the quantum arithmetic operation, and store the sum in the third register. The target state $|i\rangle|s\rangle|\sum_l|x_{il}-x_{sl}|\rangle$ can be obtained by discarding the redundant registers, where $\sum_l|x_{il}-x_{sl}| = d(\mathbf{x}_i,\mathbf{x}_s)$ is the Manhattan distance between two points $\mathbf{x}_i$ and $\mathbf{x}_s$.

We now analyze the time complexity and space complexity of Q_1. First, we should use U_f for M times. The time complexity of Hadamard gates, quantum arithmetic operation, and absolute value operation can be omitted compared with other steps. The time complexity of $\mathbf{O}_X$ is $O[\log(MN)]$, and we should query it for $2M$ times to get the state $|i\rangle|s\rangle|0\rangle^{\otimes\lceil\log q\rceil}\bigotimes_{l=1}^{M}(|l\rangle|x_{il}\rangle|x_{sl}\rangle)$. At the final step, we should perform quantum arithmetic operation for M times to obtain $d(\mathbf{x}_i,\mathbf{x}_s)$. Therefore, the total time complexity of Q_1 is $O[M\log(MN)]$.

As for the space complexity, $M(\lceil\log M\rceil+2\lceil\log q_1\rceil+3)$ auxiliary qubits are required to obtain the state $|i\rangle|s\rangle|d(\mathbf{x}_i,\mathbf{x}_s)\rangle$.

APPENDIX B: DETAILED ANALYSIS OF THE AMPLITUDE AMPLIFICATION IN STAGE (3.1)

The fixed-point quantum search algorithm [37] performs the sequence of the generalized Grover operator to amplify the success probability of a target state with an adjustable bound. It can be used as a subroutine in any scenario where amplitude amplification or Grover's search is used. The obvious advantage of it is that there is no need to hunt for the correct number of iterations as in Ref. [38], and this consequently eliminates the need to ever remake the initial state and restart the algorithm.

Indeed, in stage (3.1), after performing the fixed-point quantum search algorithm, we will get the following state:

$$\sqrt{p}|\Phi^t\rangle+\sqrt{1-p^2}|\Phi^t_\perp\rangle, \tag{B1}$$

where $|\Phi^t\rangle = \frac{1}{\sqrt{|C^t_j|}}\sum_{i\in C^t_j}|i\rangle|j\rangle|0\rangle\frac{1}{\sqrt{|C^t_j|}}\sum_{s\in C^t_j}|s\rangle|j\rangle$, $|\Phi^t_\perp\rangle$ is a garbage state that is orthogonal to $|\Phi^t\rangle$. Let p_0 be the initial probability of $|\Phi^t\rangle$ before the amplitude amplification. For a given $\sigma\in(0,1)$ and a known lower bound $p_{\min}$ of p_0, the condition $L \geqslant \frac{\log(2/\sigma)}{\sqrt{p_{\min}}}$ can ensure $p \geqslant 1-\sigma^2$, where

$L=2l+1$ and l is the number of generalized Grover iterate. See Ref. [37] for more detailed analysis of p.

For convenience, in our quantum k-medoids algorithm, we only consider the ideal case for $p=1$ because our algorithm still works in other cases. To see why this is so, here we first review the following corollary in Ref. [13].

Corollary 1. Assume that for any $j=1,2,\dots,m$, a unitary transformation

$$|j\rangle|0\rangle \mapsto |j\rangle(\sqrt{a}|y_j\rangle+\sqrt{1-|a|}|y_j^{\perp}\rangle) \tag{B2}$$

for $\frac{1}{2}<|a_0|\leqslant|a|\leqslant 1$ can be performed using Q queries, then the expected number of queries made to find $\min_j y_j$ with failure probability at most δ is bounded above by $90\sqrt{m}Q\lceil\frac{\log(\frac{81m(\log m+\gamma)}{\delta})}{2(|a_0|-\frac{1}{2})^2}\rceil$, where γ is Euler's constant.

For the case $p\neq 1$, the probability to succeed in amplitude estimation in stage (3.2) will become $p(1-\eta)$. In the fixed-point quantum search algorithm, the $p_{\min}$ can be provided by using amplitude estimation. Then, the value of p is related to the number of generalized Grover iterates, and we can perform a sufficient number of iterations [roughly $O(k)$ times is enough] to ensure $p\geqslant\frac{3}{4}$. By Ref. [38], the successful probability of the amplitude estimation is at least $\frac{8}{\pi^2}$, that is, $(1-\eta)\geqslant\frac{8}{\pi^2}$. Then

$$p(1-\eta)\geqslant\frac{6}{\pi^2}>\frac{1}{2}. \tag{B3}$$

Based on the above inequality and corollary 1, in stage (3.3) we can use the quantum minimum-finding algorithm to find the minimum average distance among $\{\frac{\sum_{s\in C_j^t}d(\mathbf{x}_i,\mathbf{x}_s)}{|C_j^t|}\}_{i\in C_j^t}$ with failure probability at most δ, and its query complexity is roughly $90\sqrt{N/k}\lceil\frac{\log(\frac{81\sqrt{N/k}(\log\sqrt{N/k}+\gamma)}{\delta})}{2(p(1-\eta)-\frac{1}{2})^2}\rceil$. By simply choosing $\delta,\eta=O(1)$, it can be reduced to $\widetilde{O}(\sqrt{N/k})$. This is consistent with the conclusion of our main text.

APPENDIX C: DETAILED PROCESS OF STAGE (3.2)

The specific process of stage (3.2) is depicted as follows.

(i) We start with the initial state

$$\frac{1}{\sqrt{|C_j^t|}}\sum_{i\in C_j^t}|i\rangle|0\rangle\frac{1}{\sqrt{|C_j^t|}}\sum_{s\in C_j^t}|s\rangle|0\rangle^{\otimes\lceil\log q\rceil}, \tag{C1}$$

and apply a Hadamard gate to the second register, then perform a controlled Q_1 with the first, third, and fourth registers as the target, conditioned on the second register $|0\rangle$. Then, we get

$$\frac{1}{\sqrt{|C_j^t|}}\sum_{i\in C_j^t}|i\rangle\frac{1}{\sqrt{2}}\left(|0\rangle\frac{1}{\sqrt{|C_j^t|}}\sum_{s\in C_j^t}|s\rangle|d(\mathbf{x}_i,\mathbf{x}_s)\rangle\right.$$
$$\left.+|1\rangle\frac{1}{\sqrt{|C_j^t|}}\sum_{s\in C_j^t}|s\rangle|0\rangle\right). \tag{C2}$$

(ii) Add an ancillary qubit, and perform $|0\rangle\langle 0|_2\otimes R_{4,5}+|1\rangle\langle 1|_2\otimes I_{4,5}$ on the second, fourth, and fifth registers to get

$$\frac{1}{\sqrt{|C_j^t|}}\sum_{i\in C_j^t}|i\rangle\frac{1}{\sqrt{2}}\left[|0\rangle\frac{1}{\sqrt{|C_j^t|}}\sum_{s\in C_j^t}|s\rangle|d(\mathbf{x}_i,\mathbf{x}_s)\rangle\left(\frac{d(\mathbf{x}_i,\mathbf{x}_s)}{q}|0\rangle\right.\right.$$
$$\left.+\sqrt{1-\left(\frac{d(\mathbf{x}_i,\mathbf{x}_s)}{q}\right)^2}|1\rangle\right)$$
$$\left.+|1\rangle\frac{1}{\sqrt{|C_j^t|}}\sum_{s\in C_j^t}|s\rangle|0\rangle^{\otimes\lceil\log q\rceil}|0\rangle\right], \tag{C3}$$

where $R_{4,5}$ is a controlled rotation operator which rotates the ancillary qubit from $|0\rangle$ to $[\frac{d(\mathbf{x}_i,\mathbf{x}_s)}{q}|0\rangle+\sqrt{1-(\frac{d(\mathbf{x}_i,\mathbf{x}_s)}{q})^2}|1\rangle]$ conditioned on $|d(\mathbf{x}_i,\mathbf{x}_s)\rangle$, $I_{4,5}$ is the identity operator acting on the fourth and fifth registers, and $q=\max_{i,s\in[N]}(d[\mathbf{x}_i,\mathbf{x}_s)]=2M\max_{i\in[N],l\in[M]}|x_{il}|$. We now undo the controlled Q_1 to uncompute the fourth register. Then, we obtain

$$\frac{1}{\sqrt{|C_j^t|}}\sum_{i\in C_j^t}|i\rangle\frac{1}{\sqrt{2}}\left[|0\rangle\frac{1}{\sqrt{|C_j^t|}}\sum_{s\in C_j^t}|s\rangle\left(\frac{d(\mathbf{x}_i,\mathbf{x}_s)}{q}|0\rangle\right.\right.$$
$$\left.\left.+\sqrt{1-\left(\frac{d(\mathbf{x}_i,\mathbf{x}_s)}{q}\right)^2}|1\rangle\right)+|1\rangle\frac{1}{\sqrt{|C_j^t|}}\sum_{s\in C_j^t}|s\rangle|0\rangle\right]. \tag{C4}$$

(iii) The above state can be rewritten as

$$\frac{1}{\sqrt{|C_j^t|}}\sum_{i\in C_j^t}|i\rangle\frac{1}{\sqrt{2}}\big(|0\rangle|\varphi_{ij}^t\rangle+|1\rangle|\varphi_j^t\rangle\big), \tag{C5}$$

where

$$|\varphi_{ij}^t\rangle:=\frac{1}{\sqrt{|C_j^t|}}\sum_{s\in C_j^t}|s\rangle\left(\frac{d(\mathbf{x}_i,\mathbf{x}_s)}{q}|0\rangle+\sqrt{1-\left(\frac{d(\mathbf{x}_i,\mathbf{x}_s)}{q}\right)^2}|1\rangle\right)$$

and

$$|\varphi_j^t\rangle:=\frac{1}{\sqrt{|C_j^t|}}\sum_{s\in C_j^t}|s\rangle|0\rangle.$$

We then perform a Hadamard gate on the second register to get

$$\frac{1}{\sqrt{|C_j^t|}}\sum_{i\in C_j^t}|i\rangle\left[\frac{1}{2}|0\rangle\big(|\varphi_{ij}^t\rangle+|\varphi_j^t\rangle\big)+\frac{1}{2}|1\rangle\big(|\varphi_{ij}^t\rangle-|\varphi_j^t\rangle\big)\right]$$
$$:=\frac{1}{\sqrt{|C_j^t|}}\sum_{i\in C_j^t}|i\rangle\big(\cos\theta_{ij1}^t|\Psi_{ij0}^t\rangle+\sin\theta_{ij1}^t|\Psi_{ij1}^t\rangle\big)$$
$$:=\frac{1}{\sqrt{|C_j^t|}}\sum_{i\in C_j^t}|i\rangle|\Psi_{ij}^t\rangle, \tag{C6}$$

where $|\Psi_{ij0}^t\rangle=|0\rangle(|\varphi_{ij}^t\rangle+|\varphi_j^t\rangle)$, $|\Psi_{ij1}^t\rangle=|1\rangle(|\varphi_{ij}^t\rangle-|\varphi_j^t\rangle)$, and $\theta_{ij1}^t\in[0,\frac{\pi}{2}]$. For a given i, if we measure the first qubit

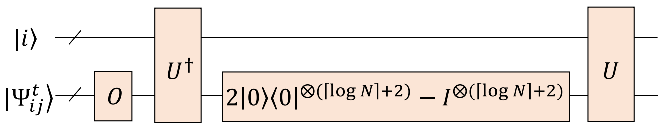
$|i\rangle$
$|\Psi^t_{ij}\rangle$
O
$U^\dagger$
$2|0\rangle\langle 0|^{\otimes(\lceil\log N\rceil+2)} - I^{\otimes(\lceil\log N\rceil+2)}$
U

FIG. 7. Quantum circuit of the Grover operator G in parallel amplitude estimation of our algorithm. Here $O = (2|0\rangle\langle 0| - I) \otimes I^{\otimes(\lceil\log N\rceil+1)}$ and U is a unitary that performs the following mapping: $\frac{1}{\sqrt{|C^t_j|}}\sum_{i\in C^t_j}|i\rangle|0\rangle^{\otimes(\lceil\log N\rceil+2)} \rightarrow \frac{1}{\sqrt{|C^t_j|}}\sum_{i\in C^t_j}|i\rangle|\Psi^t_{ij}\rangle$.

of $|\Psi^t_{ij}\rangle$, the probability of getting 1 is $P^t_{ij1} = (\sin\theta^t_{ij1})^2 = \frac{1-\langle\varphi^t_{ij}|\varphi^t_j\rangle}{2}$. It is worth noting that $\langle\varphi^t_{ij}|\varphi^t_j\rangle = \frac{\sum_{s\in C^t_j} d(\mathbf{x}_i,\mathbf{x}_s)}{q|C^t_j|}$. Once the value of $\langle\varphi^t_{ij}|\varphi^t_j\rangle$ is obtained, we can get the average distance of $\mathbf{x}_i$ to all the other points inside the cluster C^t_j. It means that we are able to calculate the average distance by estimating the value of θ^t_{ij1}.

(iv) Based on stages (i)–(iii), we first prepare the initial state

$$\frac{1}{\sqrt{|C^t_j|}}\sum_{i\in C^t_j}|i\rangle\left|\Psi^t_{ij}\right\rangle|0\rangle^{\otimes a}|0\rangle^{\otimes\lceil\log q\rceil}, \tag{C7}$$

where the value of a determines the accuracy of amplitude estimation and we discuss it in Sec. III B.

Then, we perform parallel amplitude estimation [38,39] with Grover operator G on it to estimate the value of θ^t_{ij1}, where the quantum circuit of G is shown in Fig. 7.

After parallel amplitude estimation, we obtain

$$\frac{1}{\sqrt{|C^t_j|}}\sum_{i\in C^t_j}|i\rangle\frac{1}{\sqrt{2}}\Bigg(e^{\imath\theta^t_{ij1}}\left|\Psi^t_{ij+}\right\rangle\left|\frac{\theta^t_{ij1}}{\pi}\right\rangle + e^{-\imath\theta^t_{ij1}}\left|\Psi^t_{ij-}\right\rangle\left|1-\frac{\theta^t_{ij1}}{\pi}\right\rangle\Bigg)|0\rangle^{\otimes\lceil\log q\rceil}, \tag{C8}$$

where $|\Psi^t_{ij\pm}\rangle = \frac{1}{\sqrt{2}}(|\Psi^t_{ij0}\rangle \mp \imath|\Psi^t_{ij1}\rangle)$, $\imath^2 = -1$ and $|\Psi^t_{ij}\rangle = \frac{1}{\sqrt{2}}(e^{\imath\theta^t_{ij1}}|\Psi^t_{ij+}\rangle + e^{-\imath\theta^t_{ij1}}|\Psi^t_{ij-}\rangle)$.

Finally, we perform a unitary $U_f : |x\rangle|0\rangle^{\otimes\lceil\log q\rceil} \rightarrow |x\rangle|f(x)\rangle$ on the last two registers to get

$$\frac{1}{\sqrt{|C^t_j|}}\sum_{i\in C^t_j}|i\rangle\frac{1}{\sqrt{2}}\Bigg(e^{\imath\theta^t_{ij1}}\left|\Psi^t_{ij+}\right\rangle\left|\frac{\theta^t_{ij1}}{\pi}\right\rangle + e^{-\imath\theta^t_{ij1}}\left|\Psi^t_{ij-}\right\rangle\left|1-\frac{\theta^t_{ij1}}{\pi}\right\rangle\Bigg)\left|\frac{\sum_{s\in C^t_j} d(\mathbf{x}_i,\mathbf{x}_s)}{|C^t_j|}\right\rangle, \tag{C9}$$

where $f(x) = q[1 - 2\sin^2(\pi x)]$, $x = \frac{\theta^t_{ij1}}{\pi}$ or $1 - \frac{\theta^t_{ij1}}{\pi}$. The target state can be obtained by discarding the redundant registers.

[1] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. R. Monroe, Nature (London) **536**, 63 (2016).
[2] T. Monz, D. Nigg, E. A. Martinez, M. F. Brandl, P. Schindler, R. Rines, S. X. Wang, I. L. Chuang, and R. Blatt, Science **351**, 1068 (2016).
[3] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. White, J. Mutus, A. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, and J. Martinis, Nature (London) **508**, 500 (2014).
[4] A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, Nat. Commun. **6**, 6979 (2015).
[5] X.-Y. Dong, Z. Li, and X. Wang, Sci. China Inf. Sci. **62**, 22501 (2019).
[6] Z.-Q. Li, B.-B. Cai, H.-W. Sun, H.-L. Liu, L.-C. Wan, S.-J. Qin, Q.-Y. Wen, and F. Gao, Sci. China Phys. Mech. Astron. **65**, 290311 (2022).
[7] P. Rebentrost, M. Mohseni, and S. Lloyd, Phys. Rev. Lett. **113**, 130503 (2014).
[8] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Quantum **4**, 226 (2020).
[9] R. Huang, X.-Q. Tan, and Q.-S. Xu, Neurocomputing **452**, 89 (2021).
[10] E. Aïmeur, G. Brassard, and S. Gambs, in *Proceedings of the 24th International Conference on Machine Learning*, ICML '07 (Association for Computing Machinery, New York, 2007), pp. 1–8
[11] E. Aïmeur, G. Brassard, and S. Gambs, Mach. Learn. **90**, 261 (2013).
[12] S. Lloyd, M. Mohseni, and P. Rebentrost, arXiv:1307.0411.
[13] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum Info. Comput. **15**, 316 (2015).
[14] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, C. A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete *et al.*, arXiv:1712.05771.
[15] I. Kerenidis and J. Landman, Phys. Rev. A **103**, 042415 (2021).
[16] G. Wang, Phys. Rev. A **96**, 012335 (2017).
[17] C.-H. Yu, F. Gao, and Q.-Y. Wen, IEEE Trans. Knowl. Data Eng. **33**, 858 (2021).
[18] C.-H. Yu, F. Gao, C. Liu, D. Huynh, M. Reynolds, and J. Wang, Phys. Rev. A **99**, 022301 (2019).
[19] S. Lloyd, M. Mohseni, and P. Rebentrost, Nat. Phys. **10**, 631 (2014).
[20] I. Cong and L. Duan, New J. Phys. **18**, 073011 (2016).
[21] S.-J. Pan, L.-C. Wan, H.-L. Liu, Q.-L. Wang, S.-J. Qin, Q.-Y. Wen, and F. Gao, Phys. Rev. A **102**, 052402 (2020).
[22] S.-J. Pan, L.-C. Wan, H.-L. Liu, Y.-S. Wu, S.-J. Qin, Q.-Y. Wen, and F. Gao, Chin. Phys. B **31**, 060304 (2022).
[23] Y.-M. Li, H.-L. Liu, S.-J. Pan, S.-J. Qin, F. Gao, and Q.-Y. Wen, arXiv:2206.05526v1.
[24] L.-C. Wan, C.-H. Yu, S.-J. Pan, F. Gao, Q.-Y. Wen, and S.-J. Qin, Phys. Rev. A **97**, 062322 (2018).

[25] H.-L. Liu, S.-J. Qin, L.-C. Wan, C.-H. Yu, S.-J. Pan, F. Gao, and Q.-Y. Wen, arXiv:2203.14451v1.

[26] L.-C. Wan, C.-H. Yu, S.-J. Pan, S.-J. Qin, F. Gao, and Q.-Y. Wen, Phys. Rev. A **104**, 062414 (2021).

[27] M. Guo, H. Liu, Y. Li, W. Li, F. Gao, S. Qin, and Q. Wen, Physica A **604**, 127936 (2022).

[28] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature (London) **549**, 195 (2017).

[29] V. Dunjko and H. J. Briegel, Rep. Prog. Phys. **81**, 074001 (2018).

[30] J. Han, M. Kamber, and J. Pei, in *Data Mining*, 3rd ed., The Morgan Kaufmann Series in Data Management Systems (Morgan Kaufmann, Boston, 2012), pp. 83–124.

[31] S. Lloyd, IEEE Trans. Inf. Theory **28**, 129 (1982).

[32] L. Kaufman and P. J. Rousseeuw, Clustering by means of medoids, in *Statistical Data Analysis Based on the L1–Norm and Related Methods*, editor by Y. Dodge (Amsterdam, North-Holland, 1987), pp. 405–416.

[33] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash, in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Red Hook, 2019).

[34] S. S. Zhou, T. Loke, J. A. Izaac, and J. B. Wang, Quant. Info. Proc. **16**, 82 (2017).

[35] L. Ruiz-Perez and J. C. Garcia-Escartin, Quant. Info. Proc. **16**, 152 (2017).

[36] E. Şahin, Int. J. Quantum. Inform. **18**, 2050035 (2020).

[37] T. J. Yoder, G. H. Low, and I. L. Chuang, Phys. Rev. Lett. **113**, 210501 (2014).

[38] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, Contemp. Math. **305**, 53 (2002).

[39] C.-H. Yu, F. Gao, Q.-L. Wang, and Q.-Y. Wen, Phys. Rev. A **94**, 042311 (2016).

[40] V. Giovannetti, S. Lloyd, and L. Maccone, Phys. Rev. Lett. **100**, 160501 (2008).

[41] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, arXiv:quant-ph/0001106.

[42] R. A. Servedio, in *Automata, Languages and Programming*, edited by F. Orejas, P. G. Spirakis, and J. van Leeuwen (Springer, Berlin, Heidelberg, 2001), pp. 1065 1080.

[43] C. H. Papadimitriou, SIAM J. Comput. **10**, 542 (1981).

[44] S. Dasgupta and P. M. Long, J. Comput. Syst. Sci. **70**, 555 (2005).

[45] C. Durr and P. Hoyer, arXiv:quant-ph/9607014.

[46] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, New York, 2011).

[47] H. Kim and H. Park, Bioinformatics **23**, 1495 (2007).