Quantum algorithm for association rules mining

Chao-Hua Yu,[1,2] Fei Gao,[1,*] Qing-Le Wang,[1] and Qiao-Yan Wen[1]
[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China*
(Received 6 August 2016; published 7 October 2016)

Association rules mining (ARM) is one of the most important problems in knowledge discovery and data mining. Given a transaction database that has a large number of transactions and items, the task of ARM is to acquire consumption habits of customers by discovering the relationships between itemsets (sets of items). In this paper, we address ARM in the quantum settings and propose a quantum algorithm for the key part of ARM, finding frequent itemsets from the candidate itemsets and acquiring their supports. Specifically, for the case in which there are $M_f^{(k)}$ frequent k-itemsets in the $M_c^{(k)}$ candidate k-itemsets ($M_f^{(k)} \leqslant M_c^{(k)}$), our algorithm can efficiently mine these frequent k-itemsets and estimate their supports by using parallel amplitude estimation and amplitude amplification with complexity $O(\frac{k\sqrt{M_c^{(k)}M_f^{(k)}}}{\epsilon})$, where ϵ is the error for estimating the supports. Compared with the classical counterpart, i.e., the classical sampling-based algorithm, whose complexity is $O(\frac{kM_c^{(k)}}{\epsilon^2})$, our quantum algorithm quadratically improves the dependence on both ϵ and $M_c^{(k)}$ in the best case when $M_f^{(k)} \ll M_c^{(k)}$ and on ϵ alone in the worst case when $M_f^{(k)} \approx M_c^{(k)}$.

DOI: 10.1103/PhysRevA.94.042311

I. INTRODUCTION

Quantum computing provides a paradigm that makes use of quantum mechanical principles, such as superposition and entanglement, to perform computing tasks in quantum systems (quantum computers) [1]. Just as classical algorithms run in the classical computers, a quantum algorithm is a step-by-step procedure run in the quantum computers for solving a certain problem, which, more interestingly, is expected to outperform the classical algorithms for the same problem. As of now, various quantum algorithms have been put forward to solve a number of problems faster than their classical counterparts [2], and mainly fall into one of three classes [3]. The first class is based on the quantum Fourier transformation [1], the most famous representative being Shor's algorithm [4] for large number factoring and discrete logarithm, which offers exponential speedup over the classical algorithms. The second class is represented by the Grover's quantum search [5] and its generalized version, i.e., amplitude amplification [6], both of which achieve quadratic speedup over the classical search algorithm. The third class contains the algorithms for quantum simulation [7], the original idea of which was suggested by Feynman [8] to speed up the simulation of quantum systems using quantum computers. In the past decade, quantum simulation has made great progress in efficient sparse Hamiltonian simulation [9].

However, it is a pity that no additional fundamental quantum algorithms, except the above three classes of quantum algorithms, are known. In addition to seeking new algorithms, another important direction for quantum computing is to apply known quantum algorithms to new problem areas, such as machine learning [10].

In the last one and a half decades, quantum machine learning [10] has become a booming research field and many quantum algorithms related to various machine learning problems have been proposed, such as a quantum algorithm for solving linear equations [11], quantum linear regression [12,13], quantum principal component analysis [14], quantum supervised learning (data classification) [15–18], quantum unsupervised learning (data clustering analysis) [17,19], quantum search engine ranking [20–23], quantum neural network [24], and so on. More excitingly, these algorithms to some degree have been shown to be faster than their classical counterparts. For example, under the condition that the quantum data as inputs are provided, the quantum support vector machine for big data classification exhibits exponential speedup over the classical support vector machine [17]. Furthermore, since machine learning is a crucial tool for data mining, which is a computational process of extracting valuable information from a large data set, one of the most important applications of quantum machine learning is to efficiently implement data mining tasks in quantum computers [25].

In this paper, we address association rules mining (ARM) [26], one of the most important problems in big data mining, in the quantum settings. Given a transaction database consisting of a large number of transactions and items, the task of ARM is to discover the association rules connecting two itemsets (an itemset is a set of items) A and B in the conditional implication form $A \Rightarrow B$, which implies that a customer who buys the items in A also tends to buy the items in B. The core of ARM is to mine the itemsets that frequently occur in the transactions, which entails finding out the itemsets whose supports (occurrence frequency) are not less than a prespecified threshold, i.e., frequent itemsets, from a number of candidate itemsets [26,27]. Herein we provide an efficient quantum algorithm for ARM based on the oracle accessing the database. In particular, for mining frequent k-itemsets (a k-itemset is a set of k items) from candidate k-itemsets, we first perform parallel amplitude estimation to estimate the supports of all the candidate k-itemsets. In other words, a quantum superposition state with each superposed term encoding the support of a candidate k-itemset is generated.

*gaof@bupt.edu.cn

After that, by employing the amplitude amplification, we search in the state for the candidate k-itemsets with supports not less than the threshold. We analyze the query complexity of our algorithm and it is shown that compared with the classical counterpart, i.e., the sampling-based algorithm [28], our algorithm improves the complexity at least in the dependence on the error of estimating the supports in amplitude estimation while keeping the other parameters invariant.

The rest of this paper is organized as follows. In Sec. II, we review ARM in terms of its basic concepts, notations, and classical algorithmic procedures. Section III presents the details of our quantum algorithm and provides complexity analysis on this algorithm. Discussions and conclusions are given in the last section.

II. REVIEW OF ARM

In this section, we briefly review some basic concepts and notations of ARM and classical algorithmic procedures implementing ARM. More details can be seen in Ref. [26].

A transaction database, i.e., the objective that ARM deals with, that contains N transactions can be denoted by the set $\mathcal{T} = \{T_0, T_1, \ldots, T_{N-1}\}$ and each transaction is a subset of the set of M items $\mathcal{I} = \{I_0, I_1, \ldots, I_{M-1}\}$, i.e., $T_i \subseteq \mathcal{I}$. It can also be represented by a $N \times M$ binary matrix, denoted by D, in which the element $D_{ij} = 1(0)$ means that the item I_j is (not) contained in the transaction T_i. To illustrate it, a simple example is given in Fig. 1.

A set of items is called an *itemset*. The *support* of an itemset X is defined as the proportion of transactions in $\mathcal{T}$ that contains all of the items in X, i.e., $\mathrm{supp}(X) = \frac{|\{T_i | X \subseteq T_i\}|}{N}$. An association rule is of the implication form $A \Rightarrow B$, where A and B are two disjoint itemsets. Its support is defined as $\mathrm{supp}(A \Rightarrow B) = \mathrm{supp}(A \cup B)$ and its confidence is defined as $\mathrm{conf}(A \Rightarrow B) = \frac{\mathrm{supp}(A \cup B)}{\mathrm{supp}(A)}$. A rule is called frequent (confident) if its support (confidence) is not less than a prespecified threshold min_supp (min_conf). The task of ARM is to find out the rules $A \Rightarrow B$ that are both frequent and confident. Implementing this task consists of two phases [26]:

(1) find all the frequent itemsets X, defined as $\mathrm{supp}(X) \geqslant$ min_supp;

(2) find all the confident rules $A \Rightarrow B$ such that $A \cup B = X$.

Since the second phase is much less costly than the first [26], the core work of ARM lies in the first phase. Therefore, the task of mining association rules can be reduced to that of mining frequent itemsets. In the classical regime, there are various algorithms [26] for mining frequent itemsets, the most famous one being the *Apriori* algorithm [26,27]. Based on the important *Apriori property* stating that all nonempty subsets of a frequent itemset must also be frequent, the Apriori algorithm employs an iterative approach known as a level-wise search to discover all the frequent itemsets, the whole process of which is depicted in Fig. 2. In the kth iteration of the algorithm, two procedures are executed:

(P1) Given the set of *candidate* k-itemsets $\mathcal{C}^{(k)}$ which is just $\mathcal{I}$ when $k = 1$, the supports of all the elements in $\mathcal{C}^{(k)}$ are examined by passing every transaction of the database and the frequent elements are picked out to form the set of all frequent k-itemsets $\mathcal{F}^{(k)}$. This procedure can be seen as performing a function fre_exam that finds out frequent itemsets from candidate itemsets, namely, $\mathcal{F}^{(k)} = \mathrm{fre\_exam}(\mathcal{C}^{(k)})$.

(P2) Generate the set of candidate $(k+1)$-itemsets $\mathcal{C}^{(k+1)}$ from $\mathcal{F}^{(k)}$. This procedure generally consists of two steps, i.e., the join step and the prune step [26], and can also be seen as performing a function cand_gen, namely, $\mathcal{C}^{(k+1)} = \mathrm{cand\_gen}(\mathcal{F}^{(k)})$.

In practice, in each iteration, (P1) dominates the time complexity of the whole process [28]. Therefore, how to efficiently execute (P1) of each iteration, namely, finding frequent itemsets from candidate ones, is of great importance.

Transactions	Items
T_0	Bread, Cheese, Milk
T_1	Bread, Butter
T_2	Cheese, Milk
T_3	Bread, Cheese
T_4	Cheese, Butter, Milk

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

FIG. 1. An example of a transaction database that contains five transactions $\mathcal{T} = \{T_0, T_1, T_2, T_3, T_4\}$ with each one being a subset of the set of four items $\mathcal{I} = \{I_0 = \mathrm{Bread}, I_1 = \mathrm{Cheese}, I_2 = \mathrm{Butter}, I_3 = \mathrm{Milk}\}$ and its binary matrix representation.

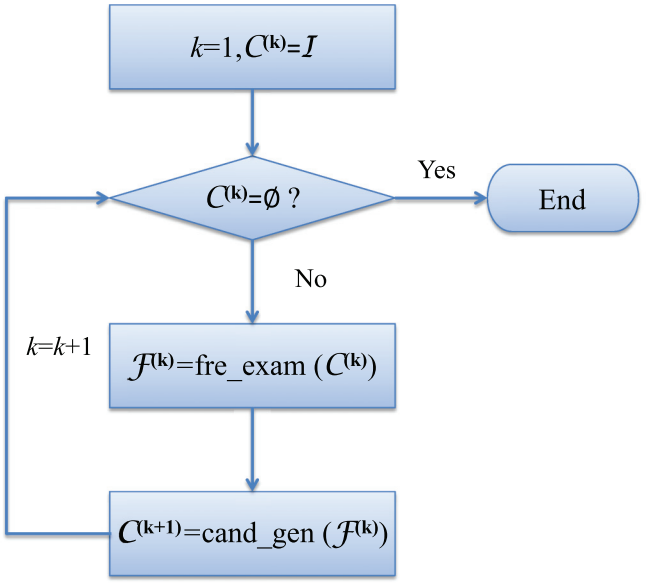

FIG. 2. The whole process of the Apriori algorithm.

In the following section, we provide a quantum algorithm to implement (P1) for each iteration, which can significantly reduce the time complexity in contrast to the classical algorithms.

III. QUANTUM ALGORITHM FOR ARM

In this section, we design a quantum algorithm based on the basic oracle O that can access the elements of the database binary matrix D. In the first place, we show how to use the basic oracle to construct oracles $O^{(k)}$ that can identify whether a transaction contains a k-itemset. Then we present in detail our algorithm that takes $O^{(k)}$ as the elementary subroutine. Finally, we analyze the query complexity of our algorithm and compare it with that of classical algorithms.

A. Constructing the oracles $O^{(k)}$ by using the basic oracle O

In our algorithm, the basic oracle O is precisely a unitary operation acting on the computational basis,

$$O|i\rangle|j\rangle|a\rangle = |i\rangle|j\rangle|a \oplus D_{ij}\rangle, \tag{1}$$

where i ranges in $\mathbb{Z}_N$ and j ranges in $\mathbb{Z}_M$. Just as with the standard Grover's algorithm [1], by taking $|a\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$, this oracle can be employed to construct a new oracle $O^{(1)}$ acting as

$$O^{(1)}|i\rangle|j\rangle = (-1)^{D_{ij}}|i\rangle|j\rangle, \tag{2}$$

which flips the phase of the state $|i\rangle|j\rangle$ when the transaction T_i contains the item I_j (i.e., $D_{ij} = 1$). Furthermore, O can be applied as the primitive to construct more complex oracles $O^{(k)}$ that can identify whether a transaction contains a k-itemset $X = \{I_{j_l} | l = 1,2,\ldots,k\}$ acting as

$$O^{(k)}|i\rangle|j_1\rangle|j_2\rangle\cdots|j_k\rangle = (-1)^{\tau(i,X)}|i\rangle|j_1\rangle|j_2\rangle\cdots|j_k\rangle, \tag{3}$$

where $\tau(i,X) = \prod_{l=1}^{k} D_{ij_l}$ is a Boolean value which identifies whether the transaction T_i contains X or, equivalently, whether $X \subseteq T_i$. That is, if $X \subseteq T_i$ [i.e., $\tau(i,X) = 1$], the phase of the state $|i\rangle|j_1\rangle|j_2\rangle\cdots|j_k\rangle$ would be flipped; otherwise, the phase would not be affected. Construction of $O^{(k)}$ requires the basic oracle O and also the generalized CNOT operation $\bigwedge_k(\sigma_x)$ (σ_x is the Pauli matrix [1]) which uses $\Theta(k)$ [29] basic one-qubit and two-qubit gates to carry out the map [30],

$$|x_1\rangle|x_2\rangle\cdots|x_k\rangle|y\rangle \mapsto |x_1\rangle|x_2\rangle\cdots|x_k\rangle|y \oplus \prod_{i=1}^{k} x_i\rangle. \tag{4}$$

The detailed process of the construction can be illustrated by the quantum circuit shown in Fig. 3 which, in fact, consists of the following steps:

(1) prepare four registers in the state $|i\rangle(|j_1\rangle|j_2\rangle\cdots|j_k\rangle)(\overbrace{|0\rangle|0\rangle\cdots|0\rangle}^{k})\frac{|0\rangle-|1\rangle}{\sqrt{2}}$;

(2) perform the operation $O_k O_{k-1}\cdots O_1$ on the state and then obtain the state $|i\rangle|j_1\rangle|j_2\rangle\cdots|j_k\rangle|D_{ij_1}\rangle|D_{ij_2}\rangle\cdots|D_{ij_k}\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, where O_l is the operation of performing the oracle O on $|i\rangle$, $|j_l\rangle$, and the lth $|0\rangle$;

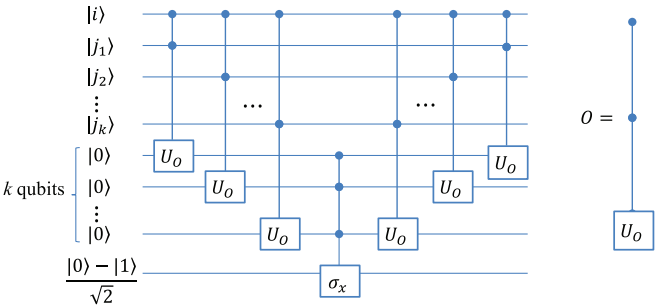

FIG. 3. The left part is the quantum circuit for constructing the oracle $O^{(k)}$ by using the basic oracle O and the generalized CNOT operation $\bigwedge_k(\sigma_x)$, where the circuit representation of O is given in the right part.

(3) apply the operation $\bigwedge_k(\sigma_x)$ to the last $k+1$ qubits and we have the state $(-1)^{\prod_{l=1}^{k} D_{ij_l}}|i\rangle|j_1\rangle|j_2\rangle\cdots|j_k\rangle|D_{ij_1}\rangle|D_{ij_2}\rangle\cdots|D_{ij_k}\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}}$;

(4) reverse step (2), discard the last $k+1$ qubits, and then implement the oracle $O^{(k)}$ as in Eq. (3).

From the above process, it is easy to see that construction of $O^{(k)}$ requires $2k$ basic oracles O and $\Theta(k)$ basic one-qubit or two-qubit gates.

B. Algorithm

Now we use the oracle $O^{(k)}$ to design our ARM algorithm to mine $\mathcal{F}^{(k)}$ from $\mathcal{C}^{(k)}$. Schematically, our algorithm will first estimate the supports of all the candidate k-itemsets (elements) in $\mathcal{C}^{(k)}$ in parallel by using amplitude estimation, and then search for candidate k-itemsets with supports not less than min_supp by employing amplitude amplification to obtain the the set of frequent k-itemsets $\mathcal{F}^{(k)}$. Here we suppose $\mathcal{C}^{(k)}$ has $M_c^{(k)}$ elements, $\mathcal{C}^{(k)} = \{C_j^{(k)} | j = 1,2,\ldots,M_c^{(k)}\}$, where $C_j^{(k)} = \{I_{c_{jl}^{(k)}} | l = 1,2,\ldots,k, c_{jl}^{(k)} \in \mathbb{Z}_M\}$, $\mathcal{F}^{(k)}$ has $M_f^{(k)}$ elements, and $\mathcal{F}^{(k)} \subseteq \mathcal{C}^{(k)}$. Mining frequent k-itemsets from $\mathcal{C}^{(k)}$ in the first place entails acquiring the supports of all the candidate k-itemsets $C_j^{(k)}$ in $\mathcal{C}^{(k)}$. Here we denote the support of $C_j^{(k)}$ by $s_j^{(k)}$. For a particular candidate k-itemset $C_j^{(k)}$, a direct method for estimating its support $s_j^{(k)}$ in a quantum computer would be the use of amplitude estimation [6].

Now we give a brief description of how quantum amplitude estimation works for estimating $s_j^{(k)}$. To achieve this task, a related oracle denoted by $O_j^{(k)}$ that should act as

$$O_j^{(k)}|i\rangle = (-1)^{\tau(i,C_j^{(k)})}|i\rangle \tag{5}$$

is required and its corresponding Grover operator is

$$G_j^{(k)} = (2|\mathcal{X}_N\rangle\langle\mathcal{X}_N| - \mathbb{I}_N)O_j^{(k)}, \tag{6}$$

where $|\mathcal{X}_N\rangle := \frac{\sum_{i=0}^{N-1}|i\rangle}{\sqrt{N}}$ and $\mathbb{I}_N$ is the identity matrix with dimension N. $G_j^{(k)}$ has two eigenvalues $\lambda_\pm = e^{\pm 2\iota\theta_j^{(k)}}$ ($\iota = \sqrt{-1}$ denotes the principal square root of -1) and corresponding eigenvectors $|\phi_{j\pm}^{(k)}\rangle$. As a matter of fact,

$$s_j^{(k)} = \sin^2\left(\theta_j^{(k)}\right). \tag{7}$$

If we initialize two registers in the state $(\frac{\sum_{t=0}^{T-1}|t\rangle}{\sqrt{T}})|\mathcal{X}_N\rangle$ and take $G_j^{(k)}$ as the Grover operator to perform amplitude (phase) estimation [6] on the state, we will finally attain the state

$$|\Phi_j^{(k)}\rangle = \frac{e^{\iota\theta_j^{(k)}}}{\sqrt{2}}\left|\mathcal{E}_T\left(\frac{\theta_j^{(k)}}{\pi}\right)\right\rangle|\phi_{j+}^{(k)}\rangle - \frac{e^{-\iota\theta_j^{(k)}}}{\sqrt{2}}\left|\mathcal{E}_T\left(1-\frac{\theta_j^{(k)}}{\pi}\right)\right\rangle|\phi_{j-}^{(k)}\rangle, \tag{8}$$

where the global phase is ignored and $|\mathcal{E}_T(\omega)\rangle = |T\omega\rangle$, when $T\omega$ is an integer, and otherwise

$$|\mathcal{E}_T(\omega)\rangle = \sum_{y=0}^{T-1}\frac{e^{2\pi\iota(T\omega-y)}-1}{T\left(e^{\frac{2\pi\iota(T\omega-y)}{T}}-1\right)}|y\rangle. \tag{9}$$

Then measuring $|\Phi_j^{(k)}\rangle$ in the computational basis in the first register will, with a high probability, output some $\tilde{y}_j$ or $T-\tilde{y}_j$ such that $\sin^2(\frac{\pi\tilde{y}_j}{T}) = \sin^2[\frac{\pi(T-\tilde{y}_j)}{T}] \approx s_j^{(k)}$; thus, $\sin^2(\frac{\pi\tilde{y}_j}{T})$ or $\sin^2[\frac{\pi(T-\tilde{y}_j)}{T}]$ can be taken as the estimate for $s_j^{(k)}$.

Surprisingly, when confining to $C_j^{(k)}$ and letting

$$|C_j^{(k)}\rangle := \otimes_{l=1}^{k}|c_{jl}^{(k)}\rangle, \tag{10}$$

$O^{(k)}$ has the same function as $O_j^{(k)}$ [shown in Eq. (5)] according to Eq. (3),

$$O^{(k)}|i\rangle|C_j^{(k)}\rangle = (-1)^{\tau(i,C_j^{(k)})}|i\rangle|C_j^{(k)}\rangle = \left(O_j^{(k)}|i\rangle\right)|C_j^{(k)}\rangle. \tag{11}$$

Therefore, based on $O^{(k)}$, we have a Grover-like operator,

$$G^{(k)} = [(2|\mathcal{X}_N\rangle\langle\mathcal{X}_N| - \mathbb{I}_N)\otimes\mathbb{I}_{M^k}]O^{(k)}, \tag{12}$$

in contrast with $G_j^{(k)}$ [Eq. (6)], where $\mathbb{I}_{M^k}$ is due to the fact that the dimension of $|C_j^{(k)}\rangle$ is M^k. Then, from Eqs. (5), (6), (11), and (12), it is easy to derive that for any integer $y>0$, we have

$$(G^{(k)})^y\left(|\mathcal{X}_N\rangle|C_j^{(k)}\rangle\right) = \left[\left(G_j^{(k)}\right)^y|\mathcal{X}_N\rangle\right]|C_j^{(k)}\rangle. \tag{13}$$

Consequently, if we take $G^{(k)}$ to perform amplitude estimation on the three-register state $(\frac{\sum_{t=0}^{T-1}|t\rangle}{\sqrt{T}})|\mathcal{X}_N\rangle|C_j^{(k)}\rangle$ instead of $(\frac{\sum_{t=0}^{T-1}|t\rangle}{\sqrt{T}})|\mathcal{X}_N\rangle$, we can finally get the state $|\Phi_j^{(k)}\rangle|C_j^{(k)}\rangle$, in contrast with $|\Phi_j^{(k)}\rangle$ [Eq. (8)]. Furthermore, if we take the superposition state $\frac{\sum_{j=1}^{M_c^{(k)}}|C_j^{(k)}\rangle}{\sqrt{M_c^{(k)}}}$ instead of $|C_j^{(k)}\rangle$ as input, we will finally obtain the state

$$|\Phi^{(k)}\rangle = \frac{\sum_{j=1}^{M_c^{(k)}}|\Phi_j^{(k)}\rangle|C_j^{(k)}\rangle}{\sqrt{M_c^{(k)}}} \tag{14}$$

because of the linearity of the unitary operator. So the estimates of all the supports $s_j^{(k)}$ are stored in the first register in parallel. We call the process in which the "big" Grover-like operator $G^{(k)}$ is taken to perform amplitude estimation *parallel amplitude estimation*.

After parallel amplitude estimation, we perform amplitude amplification on the first register of $|\Phi^{(k)}\rangle$ to search for the terms y such that $\sin^2(\frac{\pi y}{T}) \geqslant$ min_supp or $\sin^2[\frac{\pi(T-y)}{T}] \geqslant$ min_supp, so that we obtain a superposition state encoding the frequent k-itemsets in the third register and their supports in the first register.

The overall process of our quantum algorithm (QARM) for mining frequent k-itemsets is summarized by the following five steps.

Algorithm. $\mathcal{F}^{(k)} = \mathrm{QARM}(\mathcal{C}^{(k)}, G^{(k)}, k, T)$.

(1) Prepare three registers in the state

$$|\Psi_1\rangle = \left(\frac{\sum_{t=0}^{T-1}|t\rangle}{\sqrt{T}}\right)|\mathcal{X}_N\rangle\left(\frac{\sum_{j=1}^{M_c^{(k)}}|C_j^{(k)}\rangle}{\sqrt{M_c^{(k)}}}\right). \tag{15}$$

Here $\mathcal{C}^{(k)}$, the set of all the candidate k-itemsets $C_j^{(k)}$, is just $\mathcal{I}$ for $k=1$ and is obtained by executing the classical procedure (P2) $\mathcal{C}^{(k)} = \mathrm{cand\_gen}(\mathcal{F}^{(k-1)})$ with input $\mathcal{F}^{(k-1)} = \mathrm{QARM}(\mathcal{C}^{(k-1)}, G^{(k-1)}, k-1, T)$ for $k>1$. The superposition state $\frac{\sum_{j=1}^{M_c^{(k)}}|C_j^{(k)}\rangle}{\sqrt{M_c^{(k)}}}$ can be efficiently generated for $k=1$, but is suggested to be replaced by the state $\frac{\sum_{j=1}^{M_c^{(k)}}|j\rangle|C_j^{(k)}\rangle}{\sqrt{M_c^{(k)}}}$ for efficiency for $k>1$. However, for convenience, we only consider the former state in the following steps; see the last paragraph in this section for a detailed explanation.

(2) Perform the unitary operation $\sum_{y=0}^{T-1}|y\rangle\langle y|\otimes(G^{(k)})^y$ on $|\Psi_1\rangle$ and the resulting state is

$$|\Psi_2\rangle = \left[\sum_{y=0}^{T-1}|y\rangle\langle y|\otimes(G^{(k)})^y\right]|\Psi_1\rangle. \tag{16}$$

(3) Perform the inverse Fourier transformation $F_T^\dagger$ on the first register of $|\Psi_2\rangle$ and obtain

$$|\Psi_3\rangle = (F_T^\dagger\otimes\mathbb{I}_N\otimes\mathbb{I}_{M^k})|\Psi_1\rangle = |\Phi^{(k)}\rangle, \tag{17}$$

where F_T is defined by $F_T|i\rangle = \sum_{j=0}^{T-1}\frac{e^{\frac{2\pi\iota ij}{T}}|j\rangle}{\sqrt{T}}$.

(4) Search in the first register of $|\Psi_3\rangle$ for the terms y satisfying $\sin^2(\frac{\pi y}{T}) \geqslant$ min_supp or $\sin^2[\frac{\pi(T-y)}{T}] \geqslant$ min_supp by using amplitude amplification and then obtain the state

$$|\Psi_4\rangle \approx \frac{\sum_{j=1,\,\mathrm{supp}(C_j^{(k)})\geqslant\mathrm{min\_supp}}^{M_c^{(k)}}|\Phi_j^{(k)}\rangle|C_j^{(k)}\rangle}{\sqrt{M_f^{(k)}}}. \tag{18}$$

The state contains three registers holding the estimates of the supports of frequent k-itemsets, the eigenstates of the Grover operators $G_j^{(k)}$, and the frequent k-itemsets, from left to right, respectively. $|\Phi_j^{(k)}\rangle$ is seen in Eq. (8).

(5) Measure the first and third register for $O(M_f^{(k)})$ times to reveal all the $M_f^{(k)}$ frequent k-itemsets (i.e., $\mathcal{F}^{(k)}$) and their supports.

The first three steps contribute to the key part of our algorithm, parallel amplitude estimation, and the circuit for the case that T and N are powers of 2, $T=2^t$ and $N=2^n$, which is shown in Fig. 4.

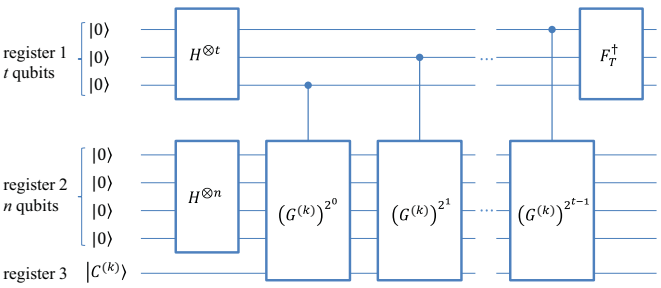

FIG. 4. Quantum circuit of the first three steps of our algorithm when $T = 2^t$ and $N = 2^n$. Here, $|C^{(k)}\rangle := \frac{\sum_{j=1}^{M_c^{(k)}} |C_j^{(k)}\rangle}{\sqrt{M_c^{(k)}}}$.

It should be stressed that when $k > 1$, it is more advisable to replace the superposition state,

$$|C^{(k)}\rangle := \frac{\sum_{j=1}^{M_c^{(k)}} |C_j^{(k)}\rangle}{\sqrt{M_c^{(k)}}}, \tag{19}$$

in step (1) of our algorithm by the two-register state,

$$|\widehat{C}^{(k)}\rangle := \frac{\sum_{j=1}^{M_c^{(k)}} |j\rangle|C_j^{(k)}\rangle}{\sqrt{M_c^{(k)}}}. \tag{20}$$

When $k = 1$, $\mathcal{C}^{(k)} = \mathcal{I}$, i.e., $C_j^{(k)} = I_{j-1}$ and $M_c^{(k)} = M$, and we can efficiently create the state $|C^{(k)}\rangle = \frac{\sum_{j=0}^{M-1} |j\rangle}{\sqrt{M}}$ in time $O[\log(M)]$. But for the case when $k > 1$, it is more desirable to use the state $|\widehat{C}^{(k)}\rangle$ instead of $|C^{(k)}\rangle$ because in this case it is probable to take less time to create $|\widehat{C}^{(k)}\rangle$ than to create $|C^{(k)}\rangle$. To generate $|\widehat{C}^{(k)}\rangle$, a quantum oracle (denoted by $O_C^{(k)}$) that performs

$$O_C^{(k)} \frac{\sum_{j=1}^{M_c^{(k)}} |j\rangle|0\rangle}{\sqrt{M_c^{(k)}}} = \frac{\sum_{j=1}^{M_c^{(k)}} |j\rangle|C_j^{(k)}\rangle}{\sqrt{M_c^{(k)}}} \tag{21}$$

is assumed to be provided in our algorithm. This oracle can be achieved via the quantum random access memory [31] in time $O[k\log(MM_c^{(k)})]$ provided by the classical data of candidate k-itemsets $C_j^{(k)}$. However, generating the state $|C^{(k)}\rangle$ from the initial k M-dimensional states $|0\rangle^{\otimes k}$ by using amplitude amplification takes time $O[k\log(M)\sqrt{\frac{M^k}{M_c^{(k)}}}]$(creating k uniform superposition states takes time $O[k\log(M)]$ and amplitude amplification takes $O(\sqrt{\frac{M^k}{M_c^{(k)}}})$ repetitions), which in practice is much more time consuming than generating $|\widehat{C}^{(k)}\rangle$. It should be noted that if the state $|\widehat{C}^{(k)}\rangle$ is taken in our algorithm, it is the state of the second register of the state, i.e., the mixed state $\frac{\sum_{j=1}^{M_c^{(k)}} |C_j^{(k)}\rangle\langle C_j^{(k)}|}{M_c^{(k)}}$ instead of the pure superposition state $|C^{(k)}\rangle$ [Eq. (19)], which will be operated in step (2) and measured in the final step.

C. Complexity analysis

In steps (1)–(3) of our algorithm, it takes $T-1$ oracles $O^{(k)}$, and the error for estimating $s_j^{(k)}$ is $\Theta[\frac{\sqrt{s_j^{(k)}(1-s_j^{(k)})}}{T}]$ [6]. Therefore, to ensure the error for estimating $s_j^{(k)}$ is $\epsilon\sqrt{s_j^{(k)}(1-s_j^{(k)})}$, T should be taken as $T = \Theta(\frac{1}{\epsilon})$. In step (4) for amplitude amplification, $O(\sqrt{\frac{M_c^{(k)}}{M_f^{(k)}}})$ repetitions (iterations) are required. The last step takes $O(M_f^{(k)})$ measurements to reveal all the $M_f^{(k)}$ frequent k-itemsets and their supports. Including all of these quantities and noting that the construction of $O^{(k)}$ entails $\Theta(k)$ basic oracles O, our algorithm takes $O(kT\sqrt{\frac{M_c^{(k)}}{M_f^{(k)}}}M_f^{(k)}) = O(\frac{k\sqrt{M_c^{(k)}M_f^{(k)}}}{\epsilon})$ basic oracles O to mine all the $M_f^{(k)}$ frequent k-itemsets ($\mathcal{F}^{(k)}$) from $M_c^{(k)}$ candidate k-itemsets ($\mathcal{C}^{(k)}$) and estimate their supports.

Now we consider the classical sampling-based algorithm for mining $\mathcal{F}^{(k)}$ from $\mathcal{C}^{(k)}$, where the supports $s_j^{(k)}$ of all the candidate k-itemsets in $\mathcal{C}^{(k)}$ are estimated by sampling the transactions of the database $\mathcal{T}$. According to the properties of binomial distribution, to ensure the induced error $\epsilon\sqrt{s_j^{(k)}(1-s_j^{(k)})}$ for estimating $s_j^{(k)}$, it needs $O(\frac{1}{\epsilon^2})$ samples to estimate every support (the same number of samples is used to estimate every support). The errors are with the same scales as those in our quantum algorithm. Since $\Theta(k)$ basic oracles O are required to identify whether a certain sample (transaction) contains an arbitrary k-itemset X, it will take $O(\frac{kM_c^{(k)}}{\epsilon^2})$ basic oracles O to estimate all the supports of $M_c^{(k)}$ candidate k-itemsets in $\mathcal{C}^{(k)}$ with precision $O(\epsilon)$. After estimating the supports by sampling, one can easily find the frequent k-itemsets and obtain their supports.

However, both of the above two algorithms are nondeterministic. If we want to mine $\mathcal{F}^{(k)}$ from $\mathcal{C}^{(k)}$ in a deterministic way, we can directly take the classical Apriori algorithm. In the algorithm, every transaction of the database is scanned to calculate the support of every candidate k-itemsets, and thus $O(kM_c^{(k)}N)$ basic oracles O are required to calculate all the supports of $M_c^{(k)}$ candidate k-itemsets and no errors are induced at all. After calculation, one can directly find the frequent k-itemsets and obtain their supports.

The comparison of our algorithm, the classical sampling-based algorithm, and the Apriori algorithm for mining $\mathcal{F}^{(k)}$ from $\mathcal{C}^{(k)}$ is given in Table I. From the comparison, two points

TABLE I. Comparisons of our quantum algorithm, the classical sampling-based algorithm, and the classical Apriori algorithm for mining $\mathcal{F}^{(k)}$ from $\mathcal{C}^{(k)}$.

Algorithm	Determinacy	Query complexity
Quantum	nondeterministic	$O(\frac{k\sqrt{M_c^{(k)}M_f^{(k)}}}{\epsilon})$
Sampling-based	nondeterministic	$O(\frac{kM_c^{(k)}}{\epsilon^2})$
Apriori	deterministic	$O(kM_c^{(k)}N)$

are derived. First, our quantum algorithm and the classical sampling-based algorithm are more efficient than the Apriori algorithm when the number of transactions N is large in most cases, but these two algorithms are nondeterministic and induce errors. So there is a trade-off between the accuracy and the complexity. Second, and more importantly, compared with the query complexity of the classical sampling-based algorithm, the query complexity of our quantum algorithm quadratically improves the dependence on the error. Since $M_f^{(k)} \leqslant M_c^{(k)}$, the improvement in the dependence on the parameter $M_c^{(k)}$ is also achieved, but the degree relies on the scale of $M_f^{(k)}$ relative to $M_c^{(k)}$. When $M_f^{(k)} \approx M_c^{(k)}$, no significant improvement is achieved. However, when $M_f^{(k)} \ll M_c^{(k)}$, quadratic improvement in the dependence on $M_c^{(k)}$ is also achieved. It is conceivable that this situation probably happens in the last iteration with setting a high minimum support threshold because (1) the number of frequent itemsets in the last iteration would be too small to generate candidate itemsets for the next iteration, and (2) a higher threshold implies a smaller number of frequent itemsets existing in candidate itemsets.

Regarding the complexity of our algorithm, two additional issues should also be addressed.

(i) *The overall query complexity of our algorithm for mining all the frequent itemsets*.

Since our quantum algorithm together with all the classical ARM algorithms finally output all the frequent itemsets instead of frequent k-itemsets for one particular k, it is necessary to analyze the overall query complexity for generating all the frequent itemsets. Assuming $\widehat{k}$ iterations are performed in the algorithm, the overall query complexity would be $O(\frac{\sum_{k=1}^{\widehat{k}} k\sqrt{M_c^{(k)} M_f^{(k)}}}{\epsilon})$, while the overall complexity of the classical sampling-based algorithm is $O(\frac{\sum_{k=1}^{\widehat{k}} k M_c^{(k)}}{\epsilon^2})$. Just as with mining frequent k-itemsets shown above, the improvement of our algorithm over the classical algorithm also consists of two parts. First, the quadratic improvement on ϵ contributed by parallel amplitude estimation is conclusive. Second, however, the improvement contributed by amplitude amplification depends on the database itself and the threshold min_supp because these two determine the sizes of $M_c^{(k)}$ and $M_f^{(k)}$. To quantify the improvement caused by amplitude amplification, we take the value

$$\gamma := \frac{\sum_{k=1}^{\widehat{k}} k M_c^{(k)}}{\sum_{k=1}^{\widehat{k}} k\sqrt{M_c^{(k)} M_f^{(k)}}}, \tag{22}$$

which means our algorithm is roughly γ times faster than the classical algorithm (regardless of the improvement caused by parallel amplitude estimation), as a measure. To show γ depends on the database itself and the threshold, two real-world transaction databases, Retail and Kosarak [32], which are usually taken to test the classical ARM algorithms, are run using the Apriori algorithm with each one taking two thresholds, 1% and 2%; see the Appendix for details. By simple calculation, we derive $\gamma = 12.75$, 25.54, 19.87, and 33.74 for the four cases: Retail 1%, Retail 2%, Kosarak 1%, and Kosarak 2%, respectively.

(ii) *The overall time complexity of invoking the operation of generating the states* $|C^{(k)}\rangle$ *and* $|\widehat{C}^{(k)}\rangle$.

According to the last paragraph of the last section, we know that in our algorithm, the states $|C^{(k)}\rangle$ and $|\widehat{C}^{(k)}\rangle$ need to be prepared for $k = 1$ and $k > 1$, respectively, and each one of the two states can be generated in time $O[k\log(M M_c^{(k)})]$ (incorporating the case $k = 1$). Taking amplitude amplification in step (4) and measurement in step (5) into consideration, the overall time complexity of invoking the operation of generating $|C^{(k)}\rangle$ and $|\widehat{C}^{(k)}\rangle$ would be $O[\log(M M_c^{(k)})k\sqrt{M_c^{(k)} M_f^{(k)}}]$. Compared with the overall query complexity of calling the basic oracles O, $O(\frac{k\sqrt{M_c^{(k)} M_f^{(k)}}}{\epsilon})$, the overall time complexity of the invocation of the operation of generating the states $|C^{(k)}\rangle$ and $|\widehat{C}^{(k)}\rangle$ is less costly. This is because, in practice, $\log(M M_c^{(k)})$ is much smaller than $\frac{1}{\epsilon}$, especially when ϵ is set to be very small ($\epsilon = 0.001$, for example). That is to say, it is the time complexity of calling the basic oracle O that dominates the overall time complexity of our quantum algorithm.

IV. CONCLUSION

In this paper, we address ARM, one of the most important problems in data mining, in the quantum settings. We provide a quantum algorithm for the core procedure of implementing ARM, mining frequent itemsets from the candidate itemsets. Specifically, by subtly using amplitude estimation and amplitude amplification, our algorithm can efficiently find the frequent k-itemsets from candidate k-itemsets and estimate their supports. Complexity analysis shows that our algorithm is faster than the classical counterpart, i.e., the classical sampling-based algorithm, in the sense that the complexity of our algorithm is at least quadratically improved in the dependence on the error. We hope our quantum algorithm for ARM can help in better understanding the power of quantum computing and inspire more quantum algorithms for big data mining tasks.

In the future, two directions of quantum ARM deserve further investigation. First, noting that our algorithm in this paper focuses on efficiently implementing the procedure (P1) mentioned in Sec. II, quantum algorithms for the procedure (P2) should be explored. Second, it is interesting to introduce privacy protection into quantum ARM. A recent work on this topic has been put forward in [33].

ACKNOWLEDGMENTS

We would like to thank the anonymous referees for their helpful comments. This work is supported by NSFC (Grants No. 61272057 and No. 61572081).

APPENDIX: THE DETAILS OF RUNNING THE CLASSICAL APRIORI ALGORITHM ON TWO REAL-WORLD TRANSACTION DATABASES

We run the Apriori algorithm [27] on two real-world transaction databases, Retail and Kosarak, which contain 88 162 transactions and 16 470 items, and 992 547 transactions and 41,270 items, respectively. We obtain the the numbers of candidate itemsets and frequent itemsets in each iteration,

TABLE II. The numbers of candidate itemsets and frequent itemsets, $M_c^{(k)}$ and $M_f^{(k)}$, for the database Retail. Here k labels the kth iteration.

	min_supp = 1%		min_supp = 2%	
k	$M_c^{(k)}$	$M_f^{(k)}$	$M_c^{(k)}$	$M_f^{(k)}$
1	16470	70	16470	20
2	2415	58	190	22
3	37	25	14	12
4	6	6	2	1

TABLE III. The numbers of candidate itemsets and frequent itemsets for the database Kosarak.

	min_supp = 1%		min_supp = 2%	
k	$M_c^{(k)}$	$M_f^{(k)}$	$M_c^{(k)}$	$M_f^{(k)}$
1	41270	54	41270	27
2	1431	140	351	45
3	194	127	45	34
4	57	52	13	13
5	11	10	2	2

which are shown in Tables II and III, for two minimum support thresholds, 1% and 2%. While five iterations are executed for Kosarak for both thresholds, four iterations are executed for Retail for both thresholds.

[1] M. A. Nielsen and I. L Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2010).

[2] A. Montanaro, Quantum algorithms: An overview, npj Quantum Inf. **2**, 15023 (2016).

[3] P. W. Shor, Why haven't more quantum algorithms been found? J. ACM **50**, 87 (2003)

[4] P. W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser (IEEE, Los Alamitos, California, 1994), pp.124–134.

[5] L. K. Grover, Quantum Mechanics Helps in Searching for a Needle in a Haystack, Phys. Rev. Lett. **79**, 325 (1997).

[6] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, *Quantum Amplitude Amplification and Estimation*, Contemporary Mathematics Series, Millenium Vol. 305 (AMS, New York, 2002).

[7] I. M. Georgescu, S. Ashhab and F. Nori, Quantum simulation, Rev. Mod. Phys. **86**, 153 (2014).

[8] R. P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. **21**, 467 (1982).

[9] D. W. Berry, A. M. Childs, and R. Kothari, Hamiltonian simulation with nearly optimal dependence on all parameters, in *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS 2015)* (IEEE, Piscataway, NJ, 2015), pp. 792–809.

[10] M. Schuld, I. Sinayskiy, and F. Petruccione, An introduction to quantum machine learning, Contemp. Phys. **56**, 172 (2015).

[11] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum Algorithm for Linear Systems of Equations, Phys. Rev. Lett. **103**, 150502 (2009).

[12] N. Wiebe, D. Braun, and S. Lloyd, Quantum Algorithm for Data Fitting, Phys. Rev. Lett. **109**, 050505 (2012).

[13] M. Schuld, I. Sinayskiy, and F. Petruccione, Prediction by linear regression on a quantum computer, Phys. Rev. A **94**, 022342 (2016).

[14] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum principal component analysis, Nat. Phys. **10**, 631 (2014).

[15] D. Anguita, S. Ridella, F. Rivieccio, and R. Zunino, Quantum optimization for training support vector machines, Neural Networks **16**, 763 (2003).

[16] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum algorithms for supervised and unsupervised machine learning, arXiv:1307.0411.

[17] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum Support Vector Machine for Big Data Classification, Phys. Rev. Lett. **113**, 130503 (2014).

[18] I. Cong and L. Duan, Quantum discriminant analysis for dimensionality reduction and classification, New J. Phys. **18**, 073011 (2016).

[19] E. Aïmeur, G. Brassard, and S. Gambs, Quantum speed-up for unsupervised learning, Machine Learning **90**, 261 (2013).

[20] S. Garnerone, P. Zanardi, and D. A. Lidar, Adiabatic Quantum Algorithm for Search Engine Ranking, Phys. Rev. Lett. **108**, 230506 (2012).

[21] G. D. Paparo and M. A. Martin-Delgado, Google in a quantum network, Sci. Rep. **2**, 444 (2012).

[22] E. Sánchez-Burillo, J. Duch, J. Gómez-Gardeñes, and D. Zueco, Quantum navigation and ranking in complex networks, Sci. Rep. **2**, 605 (2012).

[23] G. D. Paparo, M. Müller, F. Comellas, and M. A. Martin-Delgado, Quantum Google in a complex network, Sci. Rep. **3**, 2773 (2013).

[24] M. Schuld, I. Sinayskiy, and F. Petruccione, The quest for a quantum neural network, Quantum Inf. Proc. **13**, 2567 (2014).

[25] P. Wittek, *Quantum Machine Learning: What Quantum Computing Means to Data Mining* (Academic, San Diego, 2014).

[26] J. W. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed. (Morgan Kaufmann, Waltham, MA, 2011).

[27] R. Agrawal and R. Srikant, Fast algorithms for mining association rules, in *Proceedings of the 20th international conference on very large data bases* (Morgan Kaufmann, San Francisco, CA, 1994), pp. 487–499.

[28] H. Mannila, H. Toivonen, and A. I. Verkamo, Efficient algorithms for discovering association rules, in *KDD-94: AAAI workshop on Knowledge Discovery in Databases* (AAAI, Seattle, Washington, 1994), pp. 181–192.

[29] $O(f(n))$ and $\Omega(f(n))$, respectively, refer to functions that are $\leqslant c_1 f(n)$ and $\geqslant c_2 f(n)$ for some cases of $c_1, c_2 > 0$ and sufficiently large n. $\Theta(f(n))$ refers to a function that is both $O(f(n))$ and $\Omega(f(n))$.

[30] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Elementary gates for quantum computation, Phys. Rev. A **52**, 3457 (1995).

[31] V. Giovannetti, S. Lloyd, and L. Maccone, Quantum Random Access Memory, Phys. Rev. Lett. **100**, 160501 (2008).

[32] http://fimi.ua.ac.be/data/ (unpublished).

[33] S. Ying, M. Ying, and Y. Feng, Quantum privacy-preserving data mining, arXiv:1512.04009.