Journal of Quantum Information Science, 2023, 13, 1-23
https://www.scirp.org/journal/jqis
ISSN Online: 2162-576X
ISSN Print: 2162-5751

Quantum Algorithm for Mining Frequent Patterns for Association Rule Mining

Abdirahman Alasow, Marek Perkowski

Department of Electrical and Computer Engineering, Portland State University, Portland, OR, USA
Email: alasow@pdx.edu, mperkows@ee.pdx.edu

How to cite this paper: Alasow, A. and Perkowski, M. (2023) Quantum Algorithm for Mining Frequent Patterns for Association Rule Mining. *Journal of Quantum Information Science*, **13**, 1-23.
https://doi.org/10.4236/jqis.2023.131001

Received: February 21, 2023
Accepted: March 28, 2023
Published: March 31, 2023

Abstract

Maximum frequent pattern generation from a large database of transactions and items for association rule mining is an important research topic in data mining. Association rule mining aims to discover interesting correlations, frequent patterns, associations, or causal structures between items hidden in a large database. By exploiting quantum computing, we propose an efficient quantum search algorithm design to discover the maximum frequent patterns. We modified Grover's search algorithm so that a subspace of arbitrary symmetric states is used instead of the whole search space. We presented a novel quantum oracle design that employs a quantum counter to count the maximum frequent items and a quantum comparator to check with a minimum support threshold. The proposed derived algorithm increases the rate of the correct solutions since the search is only in a subspace. Furthermore, our algorithm significantly scales and optimizes the required number of qubits in design, which directly reflected positively on the performance. Our proposed design can accommodate more transactions and items and still have a good performance with a small number of qubits.

Keywords

Data Mining, Association Rule Mining, Frequent Pattern, Apriori Algorithm, Quantum Counter, Quantum Comparator, Grover's Search Algorithm

1. Introduction

We live in a digitalized era where vast amounts of data are collected daily. Data mining is a process of extracting interesting information, which is called knowledge discovery (KDD) in a database. According to [1], data mining is a field related to machine learning, but data mining is more applied than machine learn-

ing. Data mining has a broader spectrum of applications in engineering, science, business, medical applications, and many other areas. Data mining is used to process the raw data on a larger scale of data. There are various types of data mining techniques, such as association rules, classification, and clustering [2]. Association rule mining (ARM) is one of the most important research topics in data mining. ARM aims to discover interesting correlations, frequent patterns, associations, or causal structures between items hidden in a large database. Association rule mining is a branch of unsupervised learning processes that discover hidden patterns in data in the form of easily recognizable rules. Association rule mining is often termed as market basket analysis which studies the buying behaviors of customers by searching for sets of items that are frequently purchased together. Association rule mining is widely used in the retail analysis of transactions [3] [4], recommendation engines [5] [6] [7], web mining [8] [9], medical diagnosis, bioinformatics [10], and other applications [11] in various areas.

- Retail analysis of transactions: The data from past transactions can be used to generate items that the customers most like to be purchased together. The retailer can then adjust the store layout, sales strategy, bundling prices, and inventory control to take advantage of the extracted rules that are generated from association rule mining.
- Recommendation engines: Recommendation systems such as entertainment, news, and social media can be designed using association rule mining to recommend the most interesting content based on the user's past behavior.
- Web mining: Web usage mining is used in e-commerce applications to get useful information about the behavior of customers. Association rule mining is applied to the data from past behavior of the customers, and then rules based on customer preferences are generated. The developers can optimize and improve websites to personalize the web portals based on the association rules.
- Medical diagnosis: Association rule mining can be used to help diagnose patients. The symptoms of diseases and illnesses can be identified to find the conditional probability of the occurrence of illness using associate rule mining.
- Bioinformatics: There are many applications in bioinformatics problems, such as protein interaction networks, gene expression data, and others [10], that can be applied to association rule mining to identify biologically relevant patterns. These patterns can be translated into a biological context.

Association rule mining increases revenue by ensuring customer satisfaction based on customized web portals, as well as enhances medical treatment by relating the severity of the sickness and its symptoms.

Given a set of transactions in a database, the goal is to find the association rules that connect between itemsets. For example, $X \Rightarrow Y$ (X implies Y), which means the customer who buys the items in X also tends to buy the items in Y.

The implication means the co-occurrence of X and Y items. If patterns within transaction data tell us that baby formula and diapers are usually purchased together in the same transaction, a retailer can take advantage of this association for bundle pricing, product placement, and even shelf space optimization within the store layout. Association rules are considered interesting and called strong if they satisfy the user-predefined thresholds, which are minimum support (*minsup*) and minimum confidence (*minconf*) thresholds. These threshold values are pre-defined by users or domain experts.

The original association rule mining problem was first introduced in [12]. Let $I=\{I_1, I_2, \cdots, I_M\}$ be the set of all items. Let T be a set of database transactions where $T=\{T_1, T_2, \cdots, T_N\}$ and $T \subseteq I$. Each transaction is associated with an identifier, called a *TID*. A set of items is referred to as an itemset. An itemset that contains k items is called a k-itemset. k-itemsets that occur frequently are called a frequent k-itemsets. Association rules of the form $X \Rightarrow Y$ (X implies Y) is measured by support, the percentage of transactions that contain both X and Y, the union of itemsets X and Y. The support is taken to be the probability, $P(X \cup Y)$.

$$support(X \Rightarrow Y) = P(X \cup Y)$$

$$= \frac{\text{number of transactions containing both } X \text{ and } Y}{\text{Total number of transactions}}$$

$$support(X) = \frac{\text{number of transactions containing } X}{\text{Total number of transactions}}$$

The support in the above equation is called *relative support*, where the frequency or occurrence of an item is called an *absolute support* or a *support count*. Another objective measure for association rules is *confidence*, which assesses the degree of certainty of the detected association [13]. The rule $X \Rightarrow Y$ has confidence, the percentage of transactions that containing X also contains Y. The confidence is taken to be the conditional probability, $P(Y \mid X)$.

$$confidence(X \Rightarrow Y) = P(Y \mid X)$$

$$= \frac{\text{number of transactions containing } X \text{ and } Y}{\text{number of transaction containing } X} = \frac{support(X \cup Y)}{support(X)}$$

The main objective of ARM is to discover the itemsets that frequently appear in the transactions. The support (occurrence frequency) for each of these itemsets is generated from a number of candidate itemsets and not less than a pre-defined threshold [13] [14]. In **Figure 1**, association rule mining is decomposed into two phases: First, find out all the frequent itemsets such that each of these itemsets will occur at least as frequently as the pre-defined minimum support threshold. Those itemsets are called frequent or large itemsets. Second, generate association rules from those frequent itemsets with the constraints of minimum confidence threshold. There are many algorithms for mining frequent itemsets; the first phase of the association rule mining task. This first phase dominates the complexity of the whole process.

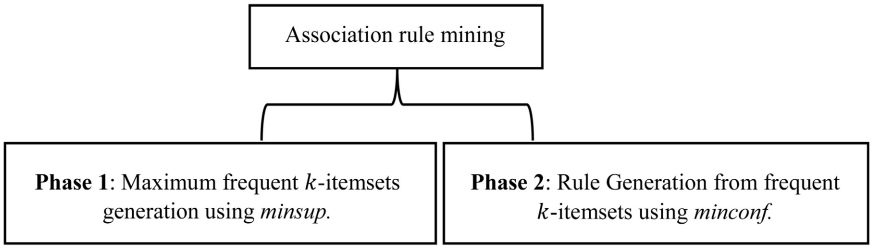

Figure 1. Association rule mining phases.

2. Related Works

2.1. Classical Algorithms for Association Rule Mining

The AIS (Agrawal, Imielinski, Swami) algorithm was the first algorithm proposed for mining association rules in [12]. In the AIS algorithm, the database was scanned many times to get maximum frequent or large itemsets. During the first pass, C_1, the candidate 1-itemsets are generated, and the support count of each individual item was accumulated. From candidate 1-itemsets, F_1 frequent 1-itemsets are generated by eliminating itemsets whose support count less than the value of *minsup*. Candidate 2-itemsets are generated by extending frequent 1-itemsets with other items in the same transaction. During the second pass over the database, the support counts of those candidate 2-itemsets are accumulated and checked against the *minsup*. Similarly, those candidate $(k+1)$-itemsets are generated by extending frequent k-itemsets with items in the same transaction. However, extending the itemsets that are not present in the previous pass results in unnecessarily generating and counting too many candidate itemsets that turn out to be small.

A number of ARM algorithms were proposed. Among these, the most well-known algorithm is the Apriori algorithm [14] that makes additional use of prune property to those candidates which have an infrequent subset before counting their supports. This optimization is possible because the support values of all subsets of a candidate are known in advance. The Apriori algorithm employs an iterative approach known as level-wise search, where k-itemsets are used to explore $(k+1)$-itemsets. First, the set of candidate 1-itemsets is found by scanning the database to accumulate the count for each item and collecting those items that satisfy the minimum support threshold. The resulting set is denoted by F_1, the set of frequent 1-itemsets. Next, F_1 is used to find C_2, the set of candidate 2-itemsets, which is used to find F_2, and so on, until no more frequent k-itemsets can be found. The finding of each C_k requires one full scan of the database, which becomes the dilemma of performance in ARM when the transaction database is very large.

In the Apriori algorithm, finding of each C_k requires one full scan of the database. To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property is used to reduce the search space. Apriori property is:

- If an itemset is frequent, then all its subsets must also be frequent.

- If an itemset is not frequent, then all its supersets cannot be frequent.

To construct candidate C_k combine frequent itemsets of size k. If $k=1$, take all 1-itemset. If $k>1$, join pairs of itemset that differ by just one item. For each generated candidate itemset ensure that all subsets of size k are frequent. The first pass of the Apriori algorithm simply counts item occurrences to determine the frequent 1-itemsets. A subsequent pass, say pass k, consists of two phases. First, the frequent itemsets F_{k-1} found in the $k-1$ pass are used to generate the candidate itemsets C_k, using two step processes: self-join and prune.

- Self-join F_{k-1}: Generate set C_k by joining F_{k-1} itemsets that share the first $k-1$ items.
- Prune: Remove from C_k the itemsets that contain a subset k-itemset that is not frequent.

Second, the database is scanned and the support of candidates in C_k is counted.

Example: Let a store promote certain computer accessories, and the store manager bundles some accessories with other discounted accessories. The manager wants to know which accessories the customers pay mostly when purchased together. Table 1 contains transactions with items.

Let *minsup* = 2 and *mincof* = 70%. We need first to generate a maximum frequent itemset. The transaction database in Table 1 is applied in the Apriori algorithm, as can be seen in Figure 2. The set of candidate 1-itemsets, C_1, obtained and then scanned the database to count the support of each itemset. The set of frequent 1-itemsets, F_1, whose support is equal to or greater than 2, the minsup value, are generated from C_1. Items in F_1 are joined to get candidate 2-itemsets, C_2. The database is scanned, and the support of each candidate itemset in C_2 is counted. The set of frequent 2-itemsets, F_2, is determined based on the support count of each candidate 2-itemset in C_2. From F_2 pairs of itemset are joined that differ by just one item to get candidate 3-itemsets, C_3. The support of each itemset

Table 1. Transaction with items.

Transaction ID	Computer accessories items
T_1	Mouse, Keyboard, Monitor, Headphone
T_2	Webcam, Monitor
T_3	Mouse, Keyboard, Monitor
T_4	Printer
T_5	Mouse, Keyboard, Headphone
T_6	Webcam, Headphone
T_7	Mouse, Monitor, Headphone
T_8	Monitor, Headphone
T_9	Mouse, Keyboard, Monitor
T_{10}	Mouse, Webcam, Monitor

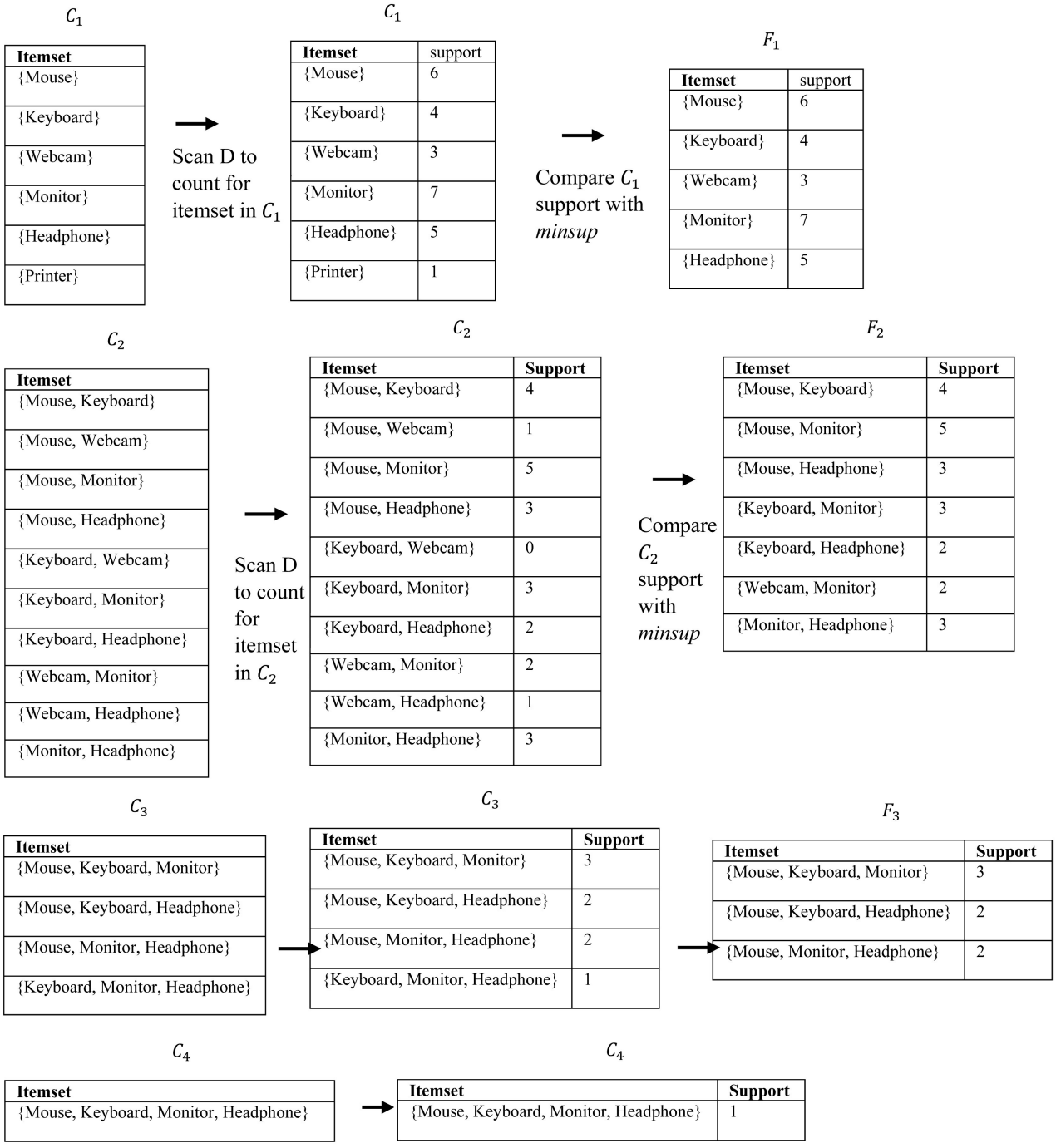

Figure 2. Generating maximum frequent itemsets using the apriori algorithm.

in C_3 is counted by scanning the database and F_3 is generated based on the min-sup value. The candidate of 4-itemset, C_4, is created by joining itemsets that differ by just two items in F_3. The support count of C_4 itemset is less than the min-sup value, so there is no frequent 4-itemsets, F_4. Thus, in this case the maximum frequent itemset is F_3. So, the maximum frequent itemset are: {Mouse, Keyboard, Monitor}, {Mouse, Keyboard, Headphone} and {Mouse, Monitor, Headphone} as can see in Figure 2.

Let generate association rules based on *mincof* = 70% for {Mouse, Keyboard, Monitor}:

Rule 1: {Mouse, Keyboard} → {Monitor}

$$confidence\left(\{\text{Mouse, Keyboard}\} \to \{\text{Monitor}\}\right) = \frac{support\{\text{Mouse, Keyboard, Monitor}\}}{support\{\text{Mouse, Keyboard}\}} = \frac{3}{4} = 75\%$$

Rule 2: {Mouse, Monitor} → {Keyboard}

$$confidence\left(\{\text{Mouse, Monitor}\} \to \{\text{Keyboard}\}\right) = \frac{support\{\text{Mouse, Keyboard, Monitor}\}}{support\{\text{Mouse, Monitor}\}} = \frac{3}{5} = 60\%$$

Rule 3: {Keyboard, Monitor} → {Mouse}

$$confidence\left(\{\text{Keyboard, Monitor}\} \to \{\text{Mouse}\}\right) = \frac{support\{\text{Mouse, Keyboard, Monitor}\}}{support\{\text{Keyboard, Monitor}\}} = \frac{3}{3} = 100\%$$

Rule 4: {Mouse} → {Keyboard, Monitor}

$$confidence\left(\{\text{Mouse}\} \to \{\text{Keyboard, Monitor}\}\right) = \frac{support\{\text{Mouse, Keyboard, Monitor}\}}{support\{\text{Mouse}\}} = \frac{3}{6} = 50\%$$

Rule 5: {Keyboard} → {Mouse, Monitor}

$$confidence\left(\{\text{Keyboard}\} \to \{\text{Mouse, Monitor}\}\right) = \frac{support\{\text{Mouse, Keyboard, Monitor}\}}{support\{\text{Keyboard}\}} = \frac{3}{4} = 75\%$$

Rule 6: {Monitor} → {Mouse, Keyboard}

$$confidence\left(\{\text{Monitor}\} \to \{\text{Mouse, Keyboard}\}\right) = \frac{support\{\text{Mouse, Keyboard, Monitor}\}}{support\{\text{Monitor}\}} = \frac{3}{7} = 42.9\%$$

If the desired *mincof* is 70%, then all interested rules that satisfy the *mincof* are rules 1, 3, and 5. Based on rule 1, 75% of the customers who purchase {Mouse, Keyboard} also purchase a Monitor. Also, based on rule 3, 100% of the customers who purchase {Keyboard, Monitor} always purchase {Mouse}.

Although the research in the Apriori algorithm has grown the recent years [15], the drawback of the Apriori algorithm is the necessity of scanning the whole database many times. Based on the Apriori algorithm, many new algorithms were designed with some modifications or improvements. Generally, there were two approaches: one is to reduce the number of passes over the whole database or replace the whole database with only part of it based on the current frequent itemsets, and another approach is to explore different kinds of pruning techniques to make the number of candidate itemsets much smaller [16]. Direct

hash and pruning (DHP) [17] and partitioning [18] are modifications of the Apriori algorithm. Another mining technique is to simultaneously mine both frequent and infrequent itemsets [19].

Associate rule mining aims to extract interesting correlations from a raw dataset that contains a huge number of database transactions and items in each transaction. Computing such a large scale of raw data in classical computing is very expensive. Leveraging the quantum search algorithm can solve such a problem significantly faster than the classical algorithm. Grover's search algorithm gives a quadratic speedup compared to an exhaustive classical algorithm for the same problem.

2.2. Quantum Algorithms for Association Rule Mining

Quantum algorithms for association rule mining [20] [21] was proposed using a quantum counting algorithm [22]. The quantum circuit design that was proposed in [20] [21] presented experimental implementation for 2 × 2 (two transactions and two items). The experiment for 2 × 2 required 9 qubits. Let T, the number of the transaction and I, the number of the items, the required number of qubits is equal to $2TI + 1$ for each iteration to find the maximum frequent k-itemset. Also, for each iteration, the whole database is a search space.

To find the maximum frequent k-itemsets from n items, we are interested only in the subspace of states of Hamming weight k. We propose an efficient method that the search space is reduced from 2^n-dimensional Hilbert space to an $\binom{n}{k}$-dimensional subspace. Also, we presented an efficient quantum circuit design that the required number of qubits reduced significantly compared with [20] [21]. We present a new quantum design method for association rule mining to generate the maximum frequent k-itemsets, which required fewer qubits, and the search space based only on the candidate k-itemset to discover the maximum frequent k-itemsets. We modified Grover's search algorithm [12] by employing the Dicke state [23] to create the superposition for k-itemset, quantum counter [24] [25] to count the frequent of k-itemset and quantum comparator to check the frequent k-itemset equal or greater than to *minsup* threshold. We start to transform the transaction database into a binary matrix such that the item value is "1" if the item is present in a transaction and "0" otherwise. The database is converted to binary matrix A_{N*M}, where each row corresponds to transaction in T, each column corresponds to an item in the set of all items I, and N is the number of transactions, and M is the number of items. A Boolean function in the sum-of-product (SOP) form is generated from the binary matrix such that each row of the matrix corresponds to one product term of the SOP function expression. In the presented case, the term is a product of variables for all items that have a value of one.

Apriori algorithm for associate rule mining, the maximum frequent k-itemset is required to scan the database for every 1-itemset, 2-itemset, etc. until to find the maximum k-itemset. The candidate itemset generated during an early itera-

tion are generally of larger magnitude than the maximum frequent *k*-itemset, which likelihood can be found in later iterations. Therefore, the initial candidate itemset generation is the key issue in improving the performance of association rule mining [17]. In this case, we started the search from the maximum *k*-itemsets as the candidate to search the maximum frequent *k*-itemsets, which may be a better chance to be the actual or close to the maximum frequent *k*-itemsets rather than starting the search from candidate 1-itemsets. Thus, the database size and the computation cost are reduced substantially. We choose only all the terms that have the maximum *k*-itemset from the SOP function. We built the quantum oracle design from the optimized SOP function that contained only the maximum *k*-itemset combined with the quantum counter and quantum comparator. We run an experiment and perform analysis using QISKIT, an IBM quantum simulator [26].

3. Quantum Oracle Design

First, let us create a binary matrix in Table 2 such that the item value is "1" if the item is present in a transaction and "0" otherwise. The binary matrix is based on items from Table 1.

To simplify, we change each item name to a letter so that we can build the SOP function. Let us observe that the SOP expressions used here are different from those applied in binary circuit synthesis. This is because here we can use repeated products or products included in other products, which can't happen in SOP expressions used for binary circuit synthesis. Therefore, our expression compiled from Table 2 is as follows:

$$abde + cd + abd + f + abe + ce + ade + de + abd + acd$$

Let *minsup* = 2 and we need to find the maximum frequent 3-itemset. First, extract all itemsets equal to or greater than the 3-itemset.

Table 2. Binary matrix corresponds from Table 1.

Transaction ID	a	b	c	d	e	f
	Mouse	Keyboard	Webcam	Monitor	Headphone	Printer
T_1	1	1		1	1	
T_2			1	1		
T_3	1	1		1		
T_4						1
T_5	1	1			1	
T_6			1		1	
T_7	1			1	1	
T_8				1	1	
T_9	1	1		1		
T_{10}	1		1	1		

$$abde + abd + abe + ade + abd + acd$$

Second, we decompose any itemset that is greater than 3-itemset. In this case, we have $abde$ which is decomposed into $abd + abe + ade + bde$ such that:

$$abd + abe + ade + bde + abd + abe + ade + abd + acd$$

Third, we build the quantum oracle for SOP function expression:

$$abd + abe + ade + bde + abd + abe + ade + abd + acd \geq 2 \quad (1)$$

Each term in the traditional quantum oracle design for the SOP function is a Toffoli gate, and the outcome is stored in an additional ancilla qubit. One large Toffoli gate that is controlled for all results is used to compute the output of the SOP function. The problem with the traditional quantum oracle architecture is that one additional ancilla qubit is needed for each term. We proposed that each term of the SOP function be connected to a quantum counter and then mirror the term back before computing the next term. In [13] [14], more information on this design is provided.

3.1. Grover's Search Algorithms

Grover's Algorithm [22] searches an unordered array of N elements to find a particular element with a given property. In classical computations, in the worst case, this search takes N queries (tests, evaluations of the classical oracle). In the average case, the particular element will be found in $N/2$ queries. Grover's algorithm can find the element in $\sqrt{N}$ queries. Thus, Grover's algorithm can be used to search the maximum frequent k-itemset for associate rule mining. Grover's algorithm is a quantum search algorithm, which speeds up a classical search algorithm of complexity $O(N)$ to $O\left(\sqrt{N}\right)$ in the space of N objects, hence Grover gives a quadratic speed up.

The SOP Boolean function in Equation (1) contains n variables from the given binary matrix in **Table 2**, which is used to represent the search space of $N = 2^n$ elements. To apply the SOP Boolean function in Grover's algorithm, these N elements are applied in a superposition state which is the input to the oracle. If the oracle recognizes an element as the solution, then the phase of the desired state is inverted. This is called the Phase inversion of the marked element. Grover's search algorithm uses another trick called inversion about the mean (average), which is also known as diffusion operation or amplitude amplification. Inversion about the mean amplifies the amplitude of the marked states and shrinks the amplitudes of other items. The amplitude amplification increases the probability of marked states, so that measuring the final states will return the target solution with a high probability near 1.

As shown in **Figure 3(a)**, the n qubits in the superposition state result from applying a vector of Hadamard gates to initial state $|0\rangle^n$. Next applied is repeated operator G which is called the Grover Loop. After the iteration of the Grover Loop operator $O\left(\sqrt{N}\right)$ times the output is measured for all input qubits. Oracle can use an arbitrary number of ancilla qubits, but all these qubits must be returned to value $|0\rangle$ inside the oracle. The number of required iterations

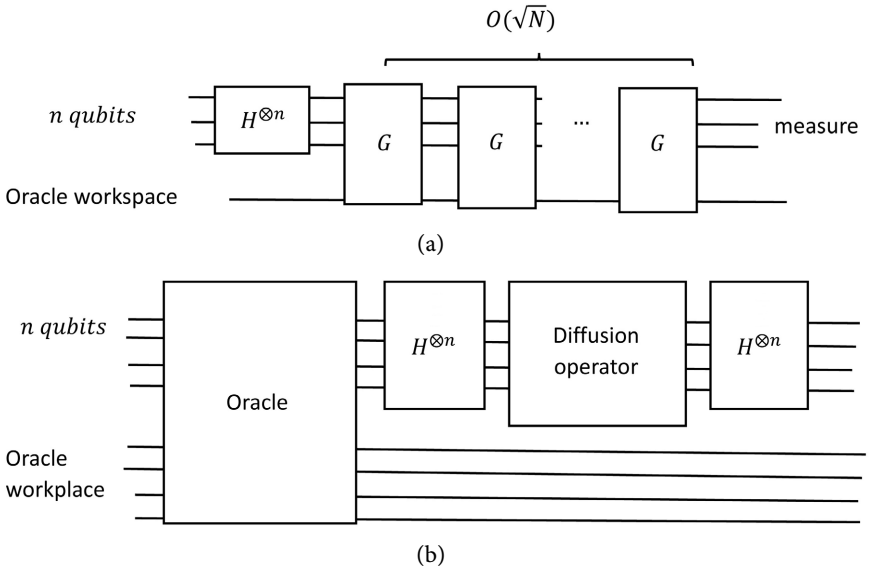

Figure 3. (a) Schematic circuit for Grover's algorithm [22]. (b) Grover operator *G*.

for the Grover operator is: $R \le \left\lceil \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rceil$ where N is number of all search space elements and M is number of solutions. The Grover Loop G is a quantum subroutine which can be broken into four steps as shown in **Figure 3(b)**.

1) Phase inversion: apply the oracle. If the oracle recognizes the solution, then the phase of the desired state is inverted.

2) Apply the Hadamard transform $H^{\otimes n}$ ($H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$).

3) Zero state phase shift: Perform the condition phase shift, in which all states receive a phase shift of -1 except for the zero state $|0\rangle$.

4) Apply the Hadamard transform $H^{\otimes n}$.

Grover's search algorithm is started to create the superposition by using Hadamard operator *H*. Hadamard operator of *n* qubits creates $N = 2^n$ quantum states. Hadamard operator of $H^{\otimes n}$ applied $|0\rangle^{\otimes n}$:

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} X\rangle$$

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H|1\rangle\frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \quad H|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$$

$$H^{\otimes 4}|0\rangle^{\otimes 4} = \frac{1}{\sqrt{16}}\left(0000\rangle + 0001\rangle + \cdots + 1111\rangle\right)$$

For maximum frequent *k*-itemsets, all N states are not solutions, but the solution would be only in Hamming weight *k*-itemsets. Hamming weight in binary is the number of ones in the binary number. *k*-itemsets is Hamming weight of *k*-itemset. For instance, finding the maximum frequent 2-itemset from 4 items, the solution will be only in 6 states with Hamming weight 2 which are equal to 0011, 0101, 0110, 1001, 1010, 1100. So, using the Hadamard operator for four

items creates 16 quantum states (0000, 0001, …, 1111) which are not required to search from the possible solutions, but the possible solutions are only in 6 quantum states of Hamming weight 2.

3.2. Dicke State

The search space of search algorithms has a critical role in terms of performance in both classical and quantum computing. In Grover's search algorithm starts preparing superposition states for the search problems. The Hadamard operator is the traditional way to create 2^n superposition states for n qubits. There are many real problems such as symmetric Boolean function [27], Johnson graph [28], and frequent patterns for associate rule mining that the possible solutions are in the form of Hamming weight, a pure symmetric state in $\binom{n}{k}$ states. For problems such as these, a proper quantum superposition state can be achieved using the Dicke state [23]. Finding the maximum frequent k-itemsets from n items, we are interested only in the subspace of states of Hamming weight k. The Dicke state $\left|D_k^n\right\rangle$ is an equal-weight superposition of all n-qubit states with Hamming Weight k (*i.e.*, all strings of length n with exactly k ones over a binary). $\left|D_k^n\right\rangle$ creates $\binom{n}{k}$ symmetric states. Below we illustrate practical examples of the Dicke state.

$$\left|D_k^n\right\rangle = \binom{n}{k}^{-\frac{1}{2}} \sum_{x\in\{0,1\}^n, wt(x)=k} X\rangle$$

where $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ and $0 < k < n$. There are n qubits and $n-k$ of them are 0 and k are 1. For instance,

$$\begin{aligned}\left|D_2^4\right\rangle &= \frac{1}{\sqrt{6}} \sum_{x\in\{0,1\}^4, wt(x)=2} X\rangle \\ &= \frac{1}{\sqrt{6}}\left(0011\rangle + 0101\rangle + 0110\rangle + 1001\rangle + 1010\rangle + 1100\rangle\right)\end{aligned}$$

$\left|D_2^4\right\rangle$ is Dicke state of 4 qubits that has 6 symmetric states and each state have 2 ones.

$$\begin{aligned}\left|D_3^5\right\rangle = \sqrt{\frac{1}{10}}\{&\left|00111\right\rangle + \left|01011\right\rangle + \left|10011\right\rangle + \left|01101\right\rangle + \left|10101\right\rangle \\ &+ \left|11001\right\rangle + \left|01110\right\rangle + \left|10110\right\rangle + \left|11010\right\rangle + \left|11100\right\rangle\}\end{aligned}$$

$\left|D_3^5\right\rangle$ is Dicke state of 5 qubits that has 10 symmetric states and each state have 3 ones.

Dicke state creates arbitrary symmetric pure states $\left|0\right\rangle^{\otimes n-k}\left|1\right\rangle^{\otimes k}$. Using the Dicke state increases the rate of the correct solution since the search is only in subspace. The classical bitstring can be converted to the Dicke state by a recursive unitary operation called Split & Cyclic Shift (SCS) unitary $SCS_{n,k}$ [23]. To

build $\left|D_k^n\right\rangle$ state, we start from $SCS_{n,k}$ where the original sequence is multiplied by a factor of $\sqrt{\frac{k}{n}}$ and then shift the first zero to the end and shift the whole sequence forward by multiplying by a factor of $\sqrt{\frac{n-k}{n}}$.

$$SCS_{n,k}:\left|0\right\rangle^{\otimes n-k}\left|1\right\rangle^{\otimes k}\to\sqrt{\frac{k}{n}}\left|0\right\rangle^{\otimes n-k}\left|1\right\rangle^{\otimes k}+\sqrt{\frac{n-k}{n}}\left|0\right\rangle^{\otimes n-k-1}\left|1\right\rangle^{\otimes k}\left|0\right\rangle$$

Dicke state given as input string $\left|0\right\rangle^{\otimes n-k}\left|1\right\rangle^{\otimes k}$ generates the entire Dicke states $\left|D_k^n\right\rangle$ by recursive unitary operation $SCS_{n,k}$.

$$SCS_{5,3}:\left|00111\right\rangle\to\sqrt{\frac{3}{5}}\left|00111\right\rangle+\sqrt{\frac{2}{5}}\left|01110\right\rangle$$

$$\sqrt{\frac{3}{5}}\left|00111\right\rangle\to\sqrt{\frac{3}{5}}\left\{\left|0011\right\rangle\right\}\otimes\left|1\right\rangle$$

$$\to\sqrt{\frac{3}{5}}\left\{\sqrt{\frac{2}{4}}\left|0011\right\rangle+\sqrt{\frac{2}{4}}\left|0110\right\rangle\right\}\otimes\left|1\right\rangle$$

$$\to\sqrt{\frac{3}{5}}\left\{\sqrt{\frac{2}{4}}\left[\sqrt{\frac{1}{3}}\left|001\right\rangle+\sqrt{\frac{2}{3}}\left|010\right\rangle\right]\otimes\left|1\right\rangle+\sqrt{\frac{2}{4}}\left[\sqrt{\frac{2}{3}}\left|011\right\rangle+\sqrt{\frac{1}{3}}\left|110\right\rangle\right]\otimes\left|0\right\rangle\right\}\otimes\left|1\right\rangle$$

$$\to\sqrt{\frac{3}{5}}\left\{\sqrt{\frac{2}{4}}\left[\sqrt{\frac{1}{3}}\left|001\right\rangle+\sqrt{\frac{2}{3}}\left(\sqrt{\frac{1}{2}}\left|01\right\rangle+\sqrt{\frac{1}{2}}\left|10\right\rangle\right)\otimes\left|0\right\rangle\right]\otimes\left|1\right\rangle+\sqrt{\frac{2}{4}}\left[\sqrt{\frac{2}{3}}\left(\sqrt{\frac{1}{2}}\left|01\right\rangle+\sqrt{\frac{1}{2}}\left|10\right\rangle\right)\otimes\left|1\right\rangle+\sqrt{\frac{1}{3}}\left|110\right\rangle\right]\otimes\left|0\right\rangle\right\}\otimes\left|1\right\rangle$$

$$\to\sqrt{\frac{3}{5}}\left\{\sqrt{\frac{2}{4}}\left[\sqrt{\frac{1}{3}}\left|001\right\rangle+\sqrt{\frac{1}{3}}\left(\left|010\right\rangle+\left|100\right\rangle\right)\right]\otimes\left|1\right\rangle+\sqrt{\frac{2}{4}}\left[\sqrt{\frac{1}{3}}\left(\left|011\right\rangle+\left|101\right\rangle\right)+\sqrt{\frac{1}{3}}\left|110\right\rangle\right]\otimes\left|0\right\rangle\right\}\otimes\left|1\right\rangle$$

$$\to\sqrt{\frac{3}{5}}\left\{\sqrt{\frac{2}{4}}\left[\sqrt{\frac{1}{3}}\left|001\right\rangle+\sqrt{\frac{1}{3}}\left|010\right\rangle+\sqrt{\frac{1}{3}}\left|100\right\rangle\right]\otimes\left|1\right\rangle+\sqrt{\frac{2}{4}}\left[\sqrt{\frac{1}{3}}\left(\left|011\right\rangle+\left|101\right\rangle\right)+\sqrt{\frac{1}{3}}\left|110\right\rangle\right]\otimes\left|0\right\rangle\right\}\otimes\left|1\right\rangle$$

$$\to\sqrt{\frac{3}{5}}\left\{\sqrt{\frac{2}{4}}\left[\sqrt{\frac{1}{3}}\left|0011\right\rangle+\sqrt{\frac{1}{3}}\left|0101\right\rangle+\sqrt{\frac{1}{3}}\left|1001\right\rangle\right]+\sqrt{\frac{2}{4}}\left[\sqrt{\frac{1}{3}}\left|0110\right\rangle+\sqrt{\frac{1}{3}}\left|1010\right\rangle+\sqrt{\frac{1}{3}}\left|1100\right\rangle\right]\right\}\otimes\left|1\right\rangle$$

$$\to\sqrt{\frac{1}{10}}\left\{\left|00111\right\rangle+\left|01011\right\rangle+\left|10011\right\rangle+\left|01101\right\rangle+\left|10101\right\rangle+\left|11001\right\rangle\right\}$$

$$\sqrt{\frac{2}{5}}|01110\rangle \to \sqrt{\frac{2}{5}}\{|0111\rangle\}\otimes|0\rangle$$

$$\to \sqrt{\frac{2}{5}}\left\{\sqrt{\frac{3}{4}}|0111\rangle+\sqrt{\frac{1}{4}}|1110\rangle\right\}\otimes|0\rangle$$

$$\to \sqrt{\frac{2}{5}}\left\{\sqrt{\frac{3}{4}}\left[\sqrt{\frac{2}{3}}|011\rangle+\sqrt{\frac{1}{3}}|110\rangle\right]\otimes|1\rangle+\sqrt{\frac{1}{4}}|1110\rangle\right\}\otimes|0\rangle$$

$$\to \sqrt{\frac{2}{5}}\left\{\sqrt{\frac{3}{4}}\left[\sqrt{\frac{2}{3}}\left(\sqrt{\frac{1}{2}}|01\rangle+\sqrt{\frac{1}{2}}|10\rangle\right)\otimes|1\rangle+\sqrt{\frac{1}{3}}|110\rangle\right]\otimes|1\rangle+\sqrt{\frac{1}{4}}|1110\right\}\otimes|0\rangle$$

$$\to \sqrt{\frac{2}{5}}\left\{\sqrt{\frac{3}{4}}\left[\sqrt{\frac{1}{3}}|011\rangle+\sqrt{\frac{1}{3}}|101\rangle+\sqrt{\frac{1}{3}}|110\rangle\right]\otimes|1\rangle+\sqrt{\frac{1}{4}}|1110\right\}\otimes|0\rangle$$

$$\to \sqrt{\frac{2}{5}}\left\{\sqrt{\frac{1}{4}}|0111\rangle+\sqrt{\frac{1}{4}}|1011\rangle+\sqrt{\frac{1}{4}}|1101\rangle+\sqrt{\frac{1}{4}}|1110\rangle\right\}\otimes|0\rangle$$

$$\to \sqrt{\frac{1}{10}}\{|01110\rangle+|10110\rangle+|11010\rangle+|11100\rangle\}$$

$$|00111\rangle \to \sqrt{\frac{1}{10}}\{|00111\rangle+|01011\rangle+|10011\rangle+|01101\rangle+|10101\rangle$$
$$+|11001\rangle+|01110\rangle+|10110\rangle+|11010\rangle+|11100\rangle\}$$

The transformation or mapping of $SCS_{n,k}$ is constructed by 1-controlled and 2-controlled Y-rotation $R_y(2\theta)$ gate between two $CNOT$, where $R_y(2\theta)=\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}$. Dicke state is constructed recursively by smaller $|D_l^n\rangle$, where $l \le k$.

$$|0\rangle^{\otimes n-k-1}|0\rangle^{\otimes k+1-l}|1\rangle^{\otimes l}$$
$$\to \sqrt{\frac{l}{n}}|0\rangle^{\otimes n-k-1}|0\rangle^{\otimes k+1-l}|1\rangle^{\otimes l}+\sqrt{\frac{n-l}{n}}|0\rangle^{\otimes n-k-1}|0\rangle^{\otimes k-l}|1\rangle^{\otimes l}|0\rangle$$

In Figure 4 and Figure 5 presented are the explicit constructions of $SCS_{n,k}$. To get more in-depth view of this construction can refer to [23]. The R_y gate is a single qubit rotation gate through angle θ radian. The R_y gate rotation is around Y-axis by angle θ. If the controlled qubits for R_y gate are all active, then the original sequence is multiplied by a factor of $\sqrt{\frac{l}{n}}$ and then shift the first zero to end and shift the whole sequence forward by multiplying by a factor of $\sqrt{\frac{n-l}{n}}$. If the controlled qubits for R_y gate are not all active, then the two $CNOT$ gates cancel each other.

The full circuit of Dicke state $|D_3^5\rangle$ is constructed recursively by $SCS_{n,k}$ as can be seen in Figure 6 which contains R_y gate in between two $CNOT$ gates. The R_y gates are controlled either one-qubit or two-qubits.

$$
\begin{aligned}
&|00\rangle_{n-1} \longrightarrow |00\rangle_{n-1}\\
&|11\rangle_{n-1} \longrightarrow |11\rangle_{n-1}\\
&|01\rangle_{n-1} \longrightarrow \sqrt{\frac{1}{n}}|01\rangle_{n-1} + \sqrt{\frac{n-1}{n}}|10\rangle_{n-1}
\end{aligned}
$$

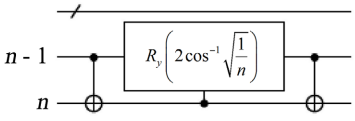

Figure 4. Construction of $SCS_{n,l}$ of a two-qubit gate [23].

$$
\begin{aligned}
&|00\rangle_{n-l}|0\rangle_n \longrightarrow |00\rangle_{n-l}|0\rangle_n\\
&|01\rangle_{n-l}|0\rangle_n \longrightarrow |01\rangle_{n-l}|0\rangle_n\\
&|00\rangle_{n-l}|1\rangle_n \longrightarrow |00\rangle_{n-l}|1\rangle_n\\
&|11\rangle_{n-l}|1\rangle_n \longrightarrow |11\rangle_{n-l}|1\rangle_n\\
&|01\rangle_{n-l}|1\rangle_n \longrightarrow \sqrt{\frac{l}{n}}|01\rangle_{n-l}|1\rangle_n + \sqrt{\frac{n-l}{n}}|11\rangle_{n-l}|0\rangle_n
\end{aligned}
$$

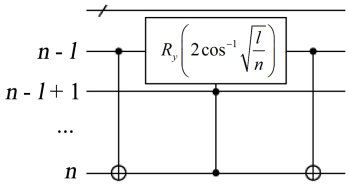

Figure 5. Construction of $SCS_{n,l}$ of a three-qubit gate [23].

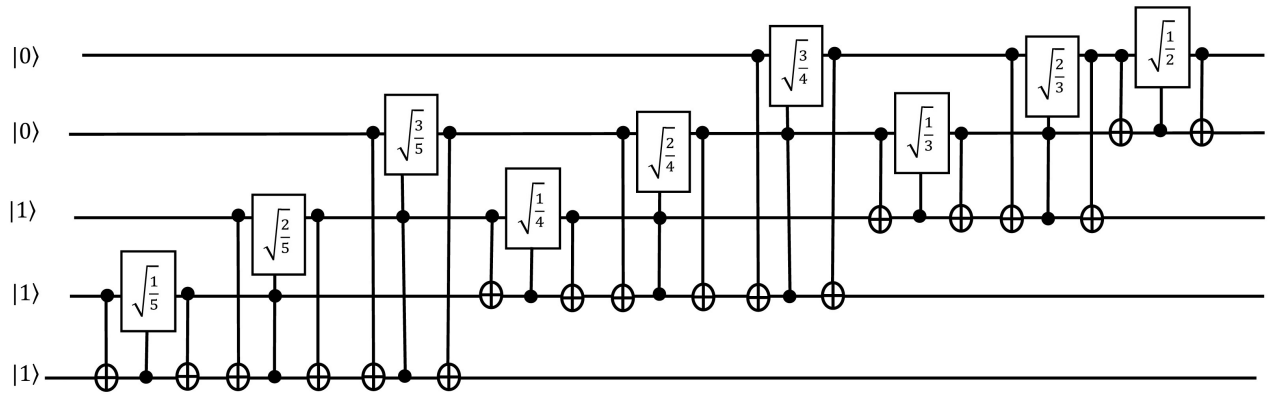

Figure 6. Dicke state $\left|D_3^5\right\rangle$ circuit, where $\sqrt{\frac{l}{n}}$ gates are shorthand for Y-roatation $R_y\left(2\cos^{-1}\sqrt{\frac{l}{n}}\right)$ [23].

3.3. Quantum Counter

Quantum counter [24] [25] is used to count the number of terms in the SOP Boolean function. Quantum counter block is a sequence of n-Toffoli followed by Feynman (*CNOT*) gates which is called Peres gate [29], as shown in **Figure 7**. The first qubit is applied a constant 1 with other variables combined, and the Peres gate is then turned into a quantum counter. (This qubit will be next taken from the term of the SOP formula to activate the counter block realized from Peres gates). For simplicity of explanation, we assume that the counter block is built from Toffoli and CNOT gates, as shown in **Figure 7**.

Here z is the least significant qubit and x the most significant. The outputs of CNOT and two of the Toffoli gates are $1 \oplus z$, $1 \cdot z \oplus y$, and $1 \cdot z \cdot y \oplus x$, respectively. When $xyz = 000$, the first Toffoli gate outputs $1 \cdot z \cdot y \oplus x = 1 \cdot 0 \cdot 0 \oplus 0 = 0 \oplus 0 = 0$ and the second $1 \cdot z \oplus y = 1 \cdot 0 \oplus 0 = 0 \oplus 0 = 0$.

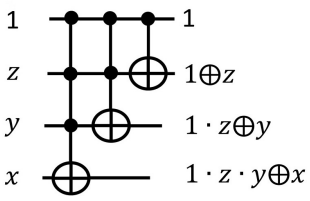

Figure 7. Three-qubit quantum counter.

The outputs of the qubits y and x are both zeros. The output of the qubit z is $1 \oplus z = 1 \oplus 0 = 1$. Hence the circuit incremented 000 by 1 to 001. Quantum counter circuit indeed outputs the value input +1.

If we connect the first control input of the quantum counter block to a circuit, then the output of the connected circuit (a term of the SOP) will either activate or deactivate the counter. When the output of the connected circuit is equal to 1, the output of the counter block is incremented by 1. When the output of the circuit is equal to 0, the output of the counter block is unchanged.

3.4. Quantum Comparator

In Table 3, the truth table of one-bit comparator which contains equal, greater than and less than.

For 1-qubit comparator, to check equal ($\overline{x_1 \oplus y_1}$) or greater than ($x_1\overline{y_1}$), we add both values $\overline{x_1 \oplus y_1} + x_1\overline{y_1}$. To build this circuit requires 5 qubits, 2 Toffoli, 4*NOT*, and 2*CNOT* gates, as can be seen in Figure 8(a). For more than 1-qubit comparator, the circuit would be huge in terms of the number of qubits and gates. To minimize the required qubits and gates, we use "not less than" such $\overline{\overline{x_1}y_1}$ which required only 3 qubits, 1 Toffoli, and 2 *NOT* gates as can be seen in Figure 8(b). The output from the quantum counter would be compared with the *minsup* value. In this case we have 4-bit quantum counter compared with *minsup* = 2 (0010). Every qubit from the quantum counter output is compared with every bit of the *minsup* value. For instance, let $x_4x_3x_2x_1$ quantum counter output and $y_4y_3y_2y_1$ *minsup* values, we build the quantum circuit for 4-bit comparator $\left(\overline{\overline{x_4}y_4}\right)\left(\overline{\overline{x_3}y_3}\right)\left(\overline{\overline{x_2}y_2}\right)\left(\overline{\overline{x_1}y_1}\right)$. In Figure 9 for 4-bit quantum comparator, we rename $x_4x_3x_2x_1$ to $counter_3counter_2counter_1counter_0$ and $y_4y_3y_2y_1$ to $comparator_3comparator_2comparator_1comparator_0$. The quantum comparator output is out_0.

In Figure 9, the four-bit comparator is compared the *minsup* value 0010 with the out of quantum counter. Every n-qubit comparator requires $3n + 1$ qubits. The n-$ancilla_i$ qubits are used to store the obtained value from the comparison.

Figure 10 is the complete quantum oracle circuit design for associate rule mining. The oracle circuit contains the circuit for each term in the SOP function expression connected with the quantum counter for each term and the quantum comparator connected with the quantum counter output. The $q_4q_3q_2q_1q_0$ represent the items and *control_counter*_0 is the control bit for the quantum

Table 3. Truth Table of one-bit comparator.

x_1y_1	$x_1 = y_1 \Rightarrow \overline{x_1 \oplus y_1}$	$x_1 > y_1 \Rightarrow x_1\overline{y_1}$	$x_1 < y_1 \Rightarrow \overline{x_1}y_1$
00	1	0	0
01	0	0	1
10	0	1	0
11	1	0	0

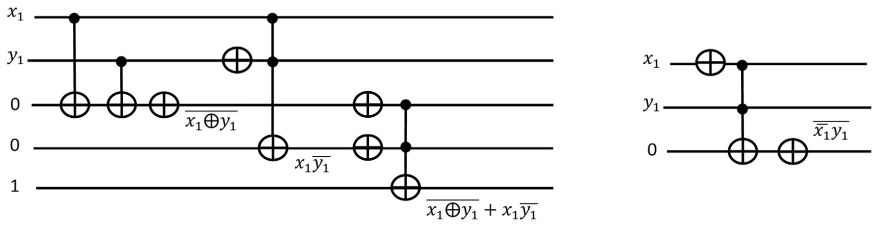

Figure 8. One-qubit comparator (a) equal or greater than. (b) Not less than.

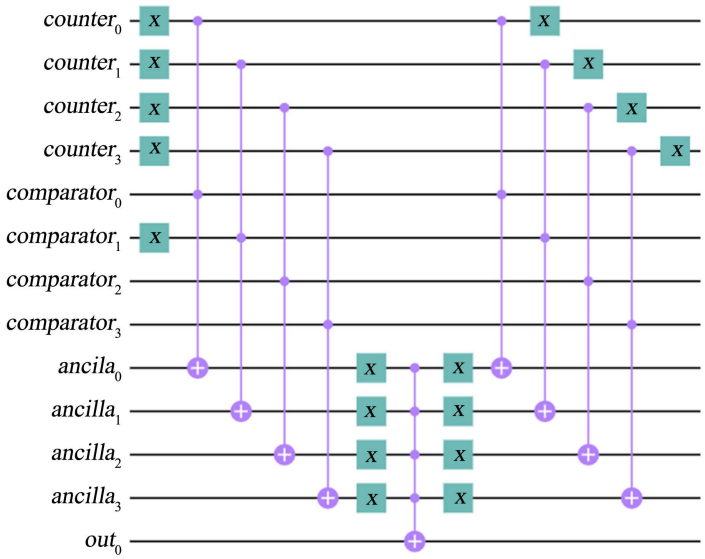

Figure 9. Four-bit quantum comparator build using IBM Qiskit simulator.

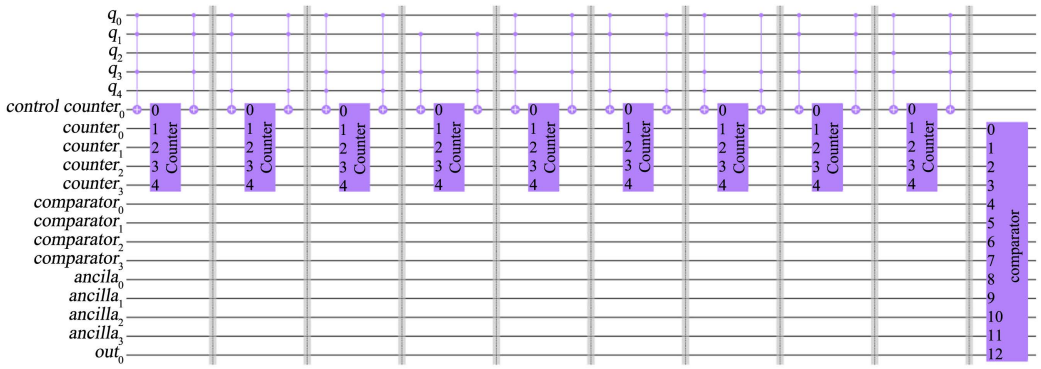

Figure 10. Full quantum oracle circuit for $abd + abe + ade + bde + abd + abe + ade + abd + acd \geq 2$.

counter. If the control bit is active (1), then the counter value is incremented by 1. If the control bit is 0, then the counter keeps its value. The number of required qubits for the quantum counter is equal to $\lceil \log_2 T \rceil + 1$, where the number of required qubits for the quantum comparator required is $3\lceil \log_2 T \rceil + 1$. Note that if $\log_2 T$ is integer value, then add 1 to the $\log_2 T$ value. The SOP function $abd + abe + ade + bde + abd + abe + ade + abd + acd \geq 2$ contains 9 terms that requires $\lceil \log_2 9 \rceil = 4$ qubits for the quantum counter. The quantum comparator compares the output of the quantum counter 4-qubit with *minsup* value 4-qubit. The quantum comparator required four additional ancilla qubits for computation and one qubit for the output. In associate rule mining, we need the maximum frequent of *k*-itemset that is equal to or greater than to *minsup* = 2. In this case, 4-qubit from the quantum counter compared with 0010. The output is out_0 is equal to 1 when frequent of *k*-itemset equal to or greater than 2.

Let us observe that the SOP function above is not used as a logical function, but it used as a pattern matching for our problem. This general idea can be used for other problems that require counting matching patterns or counting satisfied constraints. Some constraint satisfaction problems that can be formulated as such the SOP function above can refer to [30].

Figure 11 is the complete quantum algorithm circuit design for the associate rule mining, which is a modification of Grover's search algorithm. The Dicke state is used for superposition preparation, oracle, and diffuser to recognize the solution states. We applied this oracle in Grover's search algorithms for $R = 2$

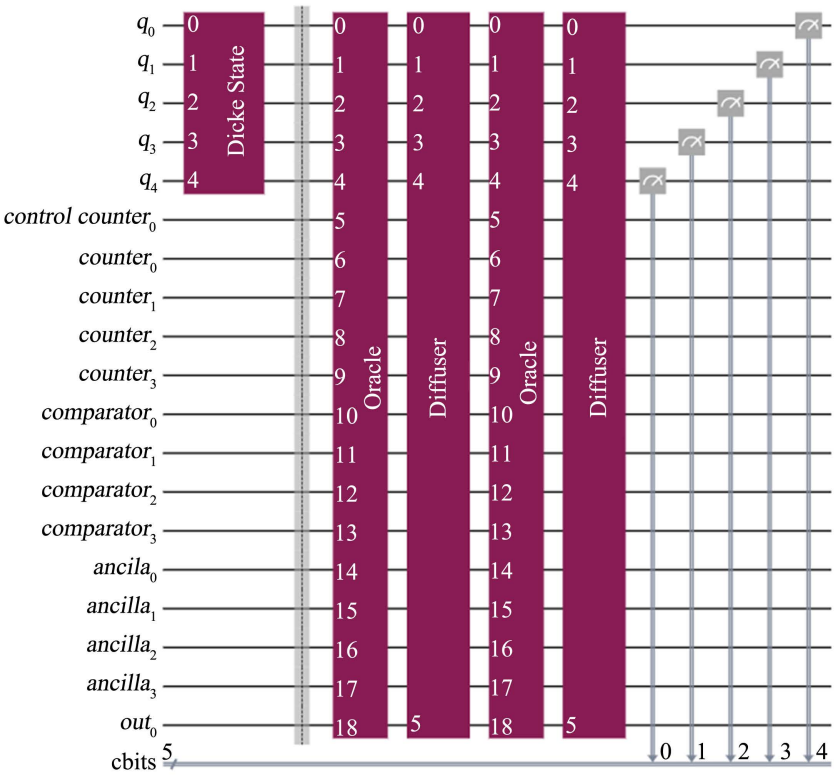

Figure 11. Full quantum algorithm circuit design for the associate rule mining.

iterations from this formula: $R \le \left\lceil \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rceil$ where $M = 3$ is the number of solutions in Figure 2 in the Apriori algorithm, and $N = 10$ is the number of all search space elements from the Dicke state of $\left|D_3^5\right\rangle$. We measured only $q_4q_3q_2q_1q_0$ for the items but for verification the measurement can be added to out_0 which is equal to 1 only if the k-itemset is greater than or equal to the *min-sup*. As can be seen in Figure 12 the values with high probability are 10,011, 11,001 and 11,010 for $abcde$ respectively based on this SOP function:

$$abd + abe + ade + bde + abd + abe + ade + abd + acd \ge 2$$

In Figure 13, we generalized our design to handle for associate rule mining to generate the maximum frequent k-itemset. As can be seen from Figure 13, the input of Dicke state is $\left|0\right\rangle^{\otimes n-k}\left|1\right\rangle^{\otimes k}$, where n is the number of items and k is the

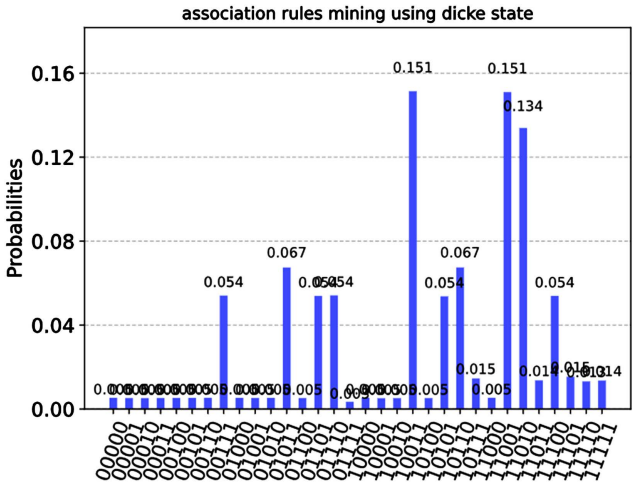

Figure 12. Histogram of measured value from Figure 11.

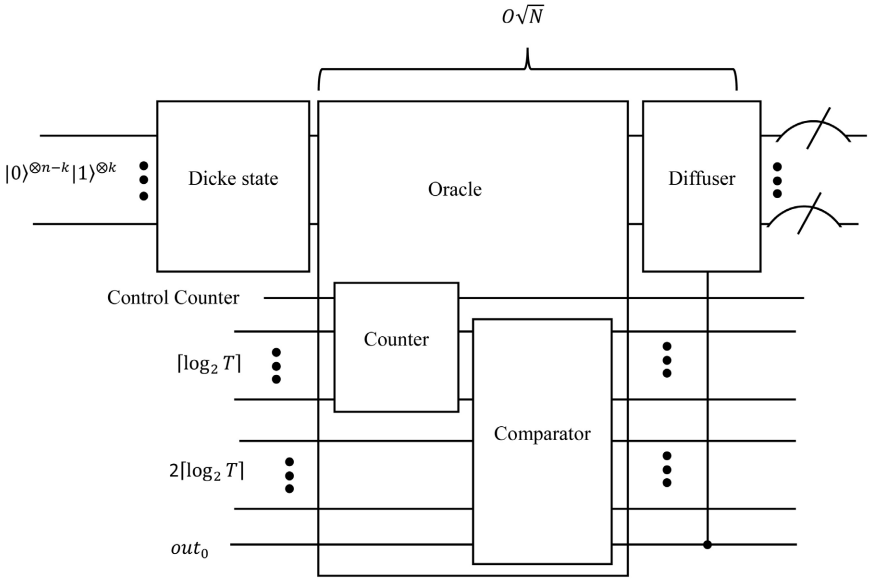

Figure 13. Proposed algorithm design for associate rule mining.

large itemset to check whether is the maximum frequent k-itemset or not. Each term from the SOP function is connected to the first qubit (Control Counter) of the quantum counter. The remaining number of qubits for the quantum counter is equal to $\lceil \log_2 T \rceil$, and the number of qubits of the quantum comparator is equal to $3\lceil \log_2 T \rceil + 1$, where T is the number of terms in the SOP function. Note that if the $\log_2 T$ is integer value, then add 1 to the $\log_2 T$ value. The out_0 is connected to the diffuser to amplify the solution. If out_0 is equal to one, then the oracle has recognized the solution. Finally, the Dicke state qubits for the items are measured. For checking purposes, the out_0 can be added to the measurement in order to check the solutions with a high probability that out_0 is equal to 1.

Scaling the required number of qubits in design is critical in the current generation of quantum computers because the number of qubits determines the degree of computational complexity. The more qubits a quantum processor possesses, the more complex and valuable the quantum circuits can run [31]. Thus, to scale and optimize the number of qubits that can accommodate in quantum algorithm design is directly reflected on the performance.

We compare the number of qubits that need in our design and the proposed design in [20] [21]. The number of required qubits in [20] [21] is equal to:

$$2TI + 1 \tag{2}$$

Our proposed design, the number of required qubits is:

$$\begin{cases} I + 3\left(\lceil \log_2 T \rceil + 1\right) + 2, & \text{If } \log_2 T \text{ is integer value} \\ I + 3\lceil \log_2 T \rceil + 2, & \text{If } \log_2 T \text{ not integer} \end{cases} \tag{3}$$

where T is the number of transactions and I is the number of items. For instance, 8 × 5 (8 transactions and 5 items), the number of required qubits in [20] [21] is equal to 81 qubits, while our design requires only 19 qubits.

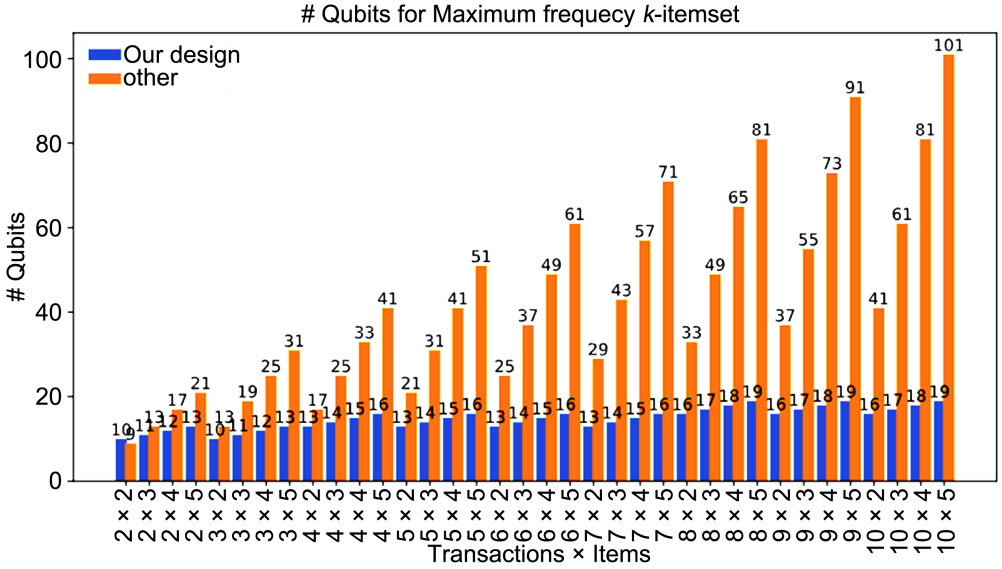

Figure 14. Histogram between our purposed design and other design in [20] [21].

As can be seen in the histogram in **Figure 14**, based on the equations in (2) and (3), for 10 transactions and 5 items, 10×5 requires 101 qubits for the design in [20] [21], while our proposed design just requires 19 qubits. According to the design in [20] [21], one qubit is needed for every item, as well as one qubit for every transaction and one qubit for the output qubit. A large transaction database typically contains massive transactions and large item sets. As a result, the number of qubits will be unreasonably high, even for large quantum computers. Our proposed quantum architecture employs the Dicke state, which reduces the search space into a sub-search space. In addition, we employ a quantum counter and a quantum comparator that can handle more transactions and items while still performing well with a small number of qubits.

4. Conclusion

We presented a novel quantum design for association rule mining to discover the maximum frequent *k*-itemset. We converted the transaction database into a new type of SOP Boolean function and then reduced the SOP function into large *k*-items with only Hamming weight that satisfies the minimum support threshold. In addition, we proposed using the Dicke state to prepare the superposition $\binom{n}{k}$ states for Grover's search algorithm instead of the conventional Hadamard operator, which would require 2^n states. We reduced the SOP function to large *k*-items where the likelihood of the maximum frequent *k*-itemset can be found. Then we design an advanced quantum oracle with a quantum counter and quantum comparator. Using these three different quantum blocks, Dicke state, quantum counter, and quantum comparator, we achieved to reduce search space and the required number of qubits in the design. We have then shown a full implementation of our design on the IBM Qiskit simulator and observed the correct results. We compared our proposal design with [20] [21] design that our design requires fewer qubits. We can solve many transactions and items by scaling and optimizing the search space and required qubits. For future improvement, a fully quantum design for association rule mining can be developed using our proposed design to generate association rules from a maximum frequent *k*-itemset.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Wittek, P. (2014) Quantum Machine Learning: What Quantum Computing Means to Data Mining. Academic Press, Cambridge.

[2] Zhao, Q. and Bhowmick, S.S. (2003) Association Rule Mining: A Survey. Nanyang Technological University, Singapore, 135.

[3] Kaur, M. and Kang, S. (2016) Market Basket Analysis: Identify the Changing Trends

of Market Data Using Association Rule Mining. *Procedia Computer Science*, **85**, 78-85. https://doi.org/10.1016/j.procs.2016.05.180

[4] Ünvan, Y.A. (2021) Market Basket Analysis with Association Rules. *Communications in Statistics- Theory and Methods*, **50**, 1615-1628. https://doi.org/10.1080/03610926.2020.1716255

[5] Yi, K., Chen, T. and Cong, G. (2018) Library Personalized Recommendation Service Method Based on Improved Association Rules. *Library Hi Tech*, **36**, 443-457. https://doi.org/10.1108/LHT-06-2017-0120

[6] Lin, W., Alvarez, S.A. and Ruiz, C. (2002) Efficient Adaptive-Support Association Rule Mining for Recommender Systems. *Data Mining and Knowledge Discovery*, **6**, 83-105. https://doi.org/10.1023/A:1013284820704

[7] Jooa, J., Bangb, S. and Parka, G. (2016) Implementation of a Recommendation System Using Association Rules and Collaborative Filtering. *Procedia Computer Science*, **91**, 944-952. https://doi.org/10.1016/j.procs.2016.07.115

[8] Langhnoja, S.G., Barot, M.P. and Mehta, D.B. (2013) Web Usage Mining Using Association Rule Mining on Clustered Data for Pattern Discovery. *International Journal of Data Mining Techniques and Applications*, **2**, 141-150.

[9] Singh, A.K., Kumar, A. and Maurya, A.K. (2014) Association Rule Mining for Web Usage Data to Improve Websites. 2014 *International Conference on Advances in Engineering & Technology Research* (*ICAETR*-2014), Unnao, 1-2 August 2014, 1-6.

[10] Naulaerts, S., Meysman, P., Bittremieux, W., Vu, T.N., Vanden Berghe, W., Goethals, B. and Laukens, K. (2015) A Primer to Frequent Itemset Mining for Bioinformatics. *Briefings in Bioinformatics*, **16**, 216-231. https://doi.org/10.1093/bib/bbt074

[11] Rajak, A. and Gupta, M.K. (2008) Association Rule Mining: Applications in Various Areas. *Proceedings of International Conference on Data Management*, Ghaziabad, India, 3-7.

[12] Agrawal, R., Imieliński, T. and Swami, A. (1993) Mining Association Rules between Sets of Items in Large Databases. *ACM SIGMOD Record*, **22**, 207-216. https://doi.org/10.1145/170036.170072

[13] Han, J., Kamber, M. and Pei, J. (2011) Data Mining: Concepts and Techniques. 3rd Edition, University of Illinois at Urbana-Champaign Micheline Kamber Jian Pei Simon Fraser University.

[14] Agrawal, R. and Srikant, R. (1994) Fast Algorithms for Mining Association Rules. *Proceedings of the* 20*th International Conference on Very Large Data Bases*, San Francisco, 12-15 September 1994, 487-499.

[15] Xie, H. (2021) Research and Case Analysis of Apriori Algorithm Based on Mining Frequent Item-Sets. *Open Journal of Social Sciences*, **9**, 458-468. https://doi.org/10.4236/jss.2021.94034

[16] Liu, H. and Wang, B. (2007) An Association Rule Mining Algorithm Based on a Boolean Matrix. *Data Science Journal*, **6**, S559-S565. https://doi.org/10.2481/dsj.6.S559

[17] Park, J.S., Chen, M.S. and Yu, P.S. (1995) An Effective Hash-Based Algorithm for Mining Association Rules. *ACM SIGMOD Record*, **24**, 175-186. https://doi.org/10.1145/568271.223813

[18] Savasere, A., Omiecinski, E.R. and Navathe, S.B. (1995) An Efficient Algorithm for Mining Association Rules in Large Databases. Georgia Institute of Technology, Atlanta.

[19] Akash, M.B., Mandal, I. and Al Mamun, M.S. (2023) Backward Support Computation Method for Positive and Negative Frequent Itemset Mining. *Journal of Data Analysis and Information Processing*, **11**, 37-48. https://doi.org/10.4236/jdaip.2023.111003

[20] Yu, C.-H., Gao, F., Wang, Q.-L. and Wen, Q.-Y. (2016) Quantum Algorithm for Association Rules Mining. *Physical Review A*, **94**, Article ID: 042311. https://doi.org/10.1103/PhysRevA.94.042311

[21] Yu, C.H. (2022) Experimental Implementation of Quantum Algorithm for Association Rules Mining. ArXiv Preprint ArXiv: 2204.13634

[22] Nielsen, M.A. and Chuang, I.L. (2002) Quantum Computation and Quantum Information; Cambridge University Press, Cambridge.

[23] Bärtschi, A. and Eidenbenz, S. (2019) Deterministic Preparation of Dicke States. In: Gąsieniec, L., Jansson, J. and Levcopoulos, C., Eds., *Fundamentals of Computation Theory. FCT* 2019. *Lecture Notes in Computer Science*, Vol. 11651, Springer, Cham, 126-139. https://doi.org/10.1007/978-3-030-25027-0_9

[24] Alasow, A., Jin, P. and Perkowski, M. (2022) Quantum Algorithm for Variant Maximum Satisfiability. *Entropy*, **24**, Article No. 1615. https://doi.org/10.3390/e24111615

[25] Alasow, A. and Perkowski, M. (2022) Quantum Algorithm for Maximum Satisfiability. 2022 *IEEE* 52*nd International Symposium on Multiple-Valued Logic* (*ISMVL*), Dallas, 18-20 May 2022, 27-34. https://doi.org/10.1109/ISMVL52857.2022.00012

[26] Aleksandrowicz, G., Alexander, T., Barkoutsos, P., Bello, L., Ben-Haim, Y., Bucher, D., Cabrera-Hernández, F.J., Carballo-Franquis, J., Chen, A., Chen, C.F., *et al.* (2019) Qiskit: An Open-Source Framework for Quantum Computing. Zenodo, Honolulu.

[27] Canteaut, A. and Videau, M. (2005) Symmetric Boolean Functions. *IEEE Transactions on Information Theory*, **51**, 2791-2811. https://doi.org/10.1109/TIT.2005.851743

[28] Wong, T.G. (2016) Quantum Walk Search on Johnson Graphs. *Journal of Physics A*: *Mathematical and Theoretical*, **49**, Article ID: 195303. https://doi.org/10.1088/1751-8113/49/19/195303

[29] Peres, A. (1985) Reversible Logic and Quantum Computers. *Physical Review A*, **32**, 3266-3276. https://doi.org/10.1103/PhysRevA.32.3266

[30] Perkowski, M. (2020) Inverse Problems, Constraint Satisfaction, Reversible Logic, Invertible Logic and Grover Quantum Oracles for Practical Problems. In: Lanese, I. and Rawski, M., Eds., *Reversible Computation. RC* 2020. *Lecture Notes in Computer Science*, Vol. 12227, Springer, Cham, 3-32. https://doi.org/10.1007/978-3-030-52482-1_1

[31] Collins, H. and Easterly, K. (2021) IBM Unveils Breakthrough 127-Qubit Quantum Processor. https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor