CBD: A Circuit-Based Database for Data Storage and Function Query in Quantum Computers

ABSTRACT

Quantum computing is a popular topic in computer science. In recent years, many new studies came out in different areas such as machine learning, network and cryptography. However, the no-cloning theorem, which states that we cannot create an independent copy of a quantum bit if we do not know the state of the source, is limiting the potential power of the quantum computer in the database area. Furthermore, superposition is the ability of a quantum system to be in multiple states simultaneously, and a measurement will destroy the superposition. Therefore, a quantum bit-based database will inevitably destroy itself when answering queries since we need to do a measurement. In this paper, we propose circuit-based databases (CBD), which use quantum circuits to store data. We use several quantum oracles (subroutines of a quantum circuit) to implement the database. The accuracy of a CBD will not deteriorate when queries are answered since the circuit will not be changed by a measurement. By quantum parallelism, it has the potential to answer multiple queries simultaneously. To answer queries to a given many-to-one function f in the CBD, we propose two algorithms. The first algorithm can answer a 1-dimensional query with $O(\sqrt{N})$ queries to the quantum oracles without extra quantum bits, where $N = 2^n$ and n is the number of quantum bits. The second algorithm can answer multiple queries with $O(\sqrt{c})$ queries to the quantum oracles in parallel, where c is the greatest number of preimages x given any $y = f(x)$. In addition, it can answer an arbitrary number of queries without any extra overhead, which can never be achieved by any classical algorithm. We conduct experiments on real-world datasets to verify the correctness of the algorithms. We consider that the CBD could be a future direction for quantum databases.

KEYWORDS

quantum algorithm, quantum database

ACM Reference Format:

. . CBD: A Circuit-Based Database for Data Storage and Function Query in Quantum Computers. In *Proceedings of* . ACM, New York, NY, USA, 14 pages.

1 INTRODUCTION

The quantum algorithm has been a widely discussed topic since Shor's algorithm was proposed in [39], which can factor an integer n in $O(polylog(n))$ time. This discovery showed the quantum algorithm can be theoretically exponentially faster than the most efficient classical algorithm in a non-trivial problem for the first time. It is believed that this algorithm can be used to break the RSA cryptosystem, which is widely used for data security. Encouraged by this discovery, many researchers studied other problems which quantum algorithms can solve more efficiently than classical algorithms.

In this paper, we focus on how to build a quantum index. Assume that we have $N = 2^n$, where N and n are two integers. Let $[N]$ denote the set of all the integer numbers from 0 to $N-1$. Given a set T of all possible sets of k-size d-dimensional points in $[N]^d$ and a function $f : [N]^d \to T$, we need to build a quantum index to store all the points and the many-to-one mapping. For simplicity, a set T is called a target set and also the k points are called the target points. Then, we need to calculate $f(x) \in T$ given an input $x \in [N]^d$. For example, given $N = 8$, $k = 3$ and $d = 1$, a target set $T = \{0, 2, 6\}$ consisting of 3 1-dimensional target points in $\{0, 1, 2, 3, 4, 5, 6, 7\}$, and a function

$$f(x) = \begin{cases} 0 & x \in \{0, 1\}; \\ 2 & x \in \{2, 3, 4, 5\}; \\ 6 & x \in \{6, 7\}, \end{cases}$$

we need to build a quantum index which we can use to calculate $f(x)$ given an input x. For example, given $x = 5$, we need to answer $f(x) = 2$. Note that all the numbers in our discussion are integers, but they can be easily extended to real numbers. To allow the input and output in a quantum superposition, by quantum parallelism, the quantum index is able to answer an arbitrary number of queries simultaneously with no extra overhead, which cannot be achieved by any classical algorithm. For example, given a superposition of $x = 5$ and $x = 7$, we need to answer a superposition of $f(x) = 2$ and $f(x) = 6$ in a single query to the quantum index. In this example, $f(x)$ is a lower-bound function. Besides the lower-bound function, it could support an arbitrary many-to-one function.

The reason why we focus on many-to-one functions is that a one-to-one function is trivial and many-to-one functions are widely-used in real-world applications. In the traditional key-value pair storage, different keys can be mapped to the same value. This many-to-one mapping can be regard as a memory that accept an input address x and return the value $f(x)$ at the address. Although different designs of a quantum RAM (QRAM) has been proposed such as the flip-flop QRAM in [34] and the bucket-brigade QRAM in [12] and they have also been widely adopted in quantum algorithms such as [21, 23, 35, 40, 41], the real-world practical implementation does not exist and the feasibility is still controversial [8]. We use c to represent the greatest number of preimages x given any $y = f(x)$. The quantum index proposed in this paper can replace the QRAMs in some applications, since the complexity only depends on c. We give two examples in the following.

- Take the nearest neighbor problem as an example, Wiebe et al. [41] proposed a quantum nearest neighbor algorithm. They assumed that the input contains a quantum oracle $\mathcal{F}$ (a

subroutine of a quantum circuit). The quantum oracle accept a vector $x = (j, l)$ and return a value $f(x)$, where $f(x)$ gives the location of the l-th non–zero entry in the j-th point. For example, given $x = (3, 2)$ and the third point $v_3 = (1, 0, 2)$, $f(x) = 3$ since the third entry is the second non-zero entry. In this problem, $c = M$ if there is M points.

- In [20], one step in the preprocessing of the product recommendation is mapping a list of purchase IDs to the list of customer IDs. The mapping from purchases to customers is obviously a many-to-one mapping. If the data is processed in quantum bits, the mapping time only depends on c, where c is the greatest number of purchases made by one customers. However, the classical mapping need at least $O(M)$ time, where M is the total number of purchases made in the supermarket. In our common knowledge, a supermarket can make tens of thousands transactions per day, but one customer usually does not make more than ten purchases in a supermarket in one day. Since M could be arbitrarily large than c, the quantum mapping could be extremely faster than the classical mapping.

Besides the application as a QRAM, the quantum index can also store other given many-to-one mappings. In classification, the nearest neighbor classifier can be regarded as a function f where x is the query point and $f(x)$ is the nearest neighbor of x. Since different nodes will have the same nearest node, f is also a many-to-one mapping, so we can store the nearest neighbor mapping in the quantum index which could accept a superposition of points and return a superposition of their nearest neighbors. In clustering, many points need to be assigned to clusters, which is also a many-to-one function f, where x is the query point and $f(x)$ is the cluster of the point. In machine learning, dimensionality reduction is very important. Since dimensionality reduction reduces cardinality, it can also be regarded as a many-to-one function f, where x is the high-dimensional point and $f(x)$ is the low-dimensional point after dimensionality reduction. By storing these mappings in a quantum index, we can do mappings for an arbitrary number of points in a single query to the quantum index.

Since the number of quantum states increases exponentially when the number of quantum bits is increasing linearly, the quantum computer is supposed to be more powerful to help a lot of calculations. In recent years, studies like [19, 26, 27] in the machine learning field are looking for applications of quantum models. To use variational quantum algorithms, where only the models are running on quantum computers and the optimizers are classical, becomes a popular direction in ML. However, such voices are relatively low in our database area, since there are still many difficulties in the way of obtaining a quantum database, according to our best knowledge:

- The first problem is the input problem shown in [7, 25, 43]. We cannot encode data in quantum bits as simple as how we store data in a classical computer, since the operations on quantum bits are limited by quantum properties.
- The second problem is the output problem described in [3, 8, 45]. Although we know that an exponential number of states can be stored in quantum bits, only one of them can be actually read out from the quantum bits due to physical properties, and all the other states will remain unknown and be destroyed. This phenomenon shows two observations. The first is that the answer to a database query should be simple since we can only keep one state out of an exponential number of states. The second is that we cannot read data from quantum bits as what we do in a classical computer, since this read operation will change the data itself simultaneously.
- The third problem is the copy problem, which is most important problem in our database area. No-cloning theorem [42] states that we cannot create an independent copy of an arbitrary quantum bit. Note that it is a basic operation in a database query to create an independent copy of an arbitrary record, where 'independent' means a change of the copy will not reflect on the original record and 'arbitrary' means the record is unknown before the query is answered.

For example, R.M. de Wolf [9] described how to use quantum fingerprints to build a quantum data structure. However, this quantum data structure can only ensure that the first M queries can obtain the correct answers with probability $\frac{2}{3}$, where M is a user parameter. The accuracy of the answer to the i-th query is $(1 - \frac{N-1}{4NM-(i-1)(n-1)})$ in the worst case, and thus the data structure could be destroyed after $4M + 1$ queries in the worst case. In many real-world applications, we cannot accept that the data will disappear by itself. The main reason of this limitation is the same as the above: The measurement for each query will change the states of the data quantum bits and disturb the accuracy of the next answer.

After considering the above three problems, we propose a new framework for a quantum database called *circuit-based databases* (CBD). In the classical computer, we are familiar with storing data in bits and proposed a lot of bit-based data structures. However, the same idea is not useful in the quantum computer, since it is problematic to store data in quantum bits as above mentioned. To tackle this problem, we introduce how to store data in a quantum circuit and answer queries with circuit-based databases. Unlike storing data in quantum bits, the accuracy of circuit-based data structures will not deteriorate with consecutive queries.

In this paper, we present how to design a CBD to store the points and a many-to-one function. We divide the information in the input and store them in quantum oracles. Then, we propose two algorithms to answer queries to the many-to-one function based on these quantum oracles. The first quantum algorithm can handle 1-dimensional cases with only the input quantum bits, i.e., it does not need any extra quantum bit. It can answer the query with $O(\sqrt{N})$ queries to the quantum oracles. The second quantum algorithm can handle d-dimensional cases with dn extra quantum bits. It can answer the query with $O(\sqrt{c})$ queries to the quantum oracles, where c is the greatest number of preimages x given any $y = f(x)$. In addition, the second algorithm can answer multiple queries in parallel with no extra overheads, which cannot be achieved by any classical algorithm. Then, we write a program to simulate the quantum process and verify the correctness of our algorithms with real-world datasets.

There are many existing bit-based classical data structures such as B-trees [1], R-trees [16] and kd-trees [2]. To use these classical data structures in our problem, they need to store all the k target

points so that they will use $O(dnk)$ bits. The CBD only cost $O(dn)$ quantum bits to obtain the answer. Since $k = O(2^n)$, the number of bits used is exponentially improved by CBD. The same comparison has been done in [30]. In addition, when answering M queries, the classical data structures need to solve them one by one, so they usually take M times as long, but the CBD can answer them simultaneously in $O(\sqrt{c})$ time with no any extra overhead compared to answering a single query. Furthermore, compared to existing bit-based quantum data structures such as the fingerprint data structure in [9], the CBD will not destroy itself by the destruction of quantum superposition.

In summary, our contributions are shown in the following:

- We are the first to propose circuit-based databases (CBD), which store data in quantum circuits instead of bits as the classical databases. The motivation is that a bit-based database will inevitably destroy itself when answering queries.
- We propose how to store data and a many-to-one function with quantum oracles in a CBD so that we can calculate the function based on the data stored.
- We propose a quantum algorithm that can answer a 1D query with $O(\sqrt{N})$ queries to the quantum oracles. It does not need any extra quantum bit, so the number of bits used is exponentially improved compared to classical data structures.
- We propose a quantum algorithm that can answer a high-dimensional query $O(\sqrt{c})$ queries to the quantum oracles, where c is the greatest number of preimages x given any $y = f(x)$. This algorithm can answer multiple queries in parallel with no extra overhead, which shows the advantage of quantum parallelism and cannot be achieved by any classical algorithm.
- We conduct experiments to verify the correctness on real-world datasets.

The rest of the paper is organized as follows. In Section 2, we introduce some basic knowledge used in this paper about quantum algorithms. In Section 3, we introduce some related works in quantum algorithms, where some are the related works about database searching in history and some are the related works in other research areas in recent years. In Section 4, we define the problem and further discuss the problem. In Section 5, we show how to build a CBD and two algorithms to answer the queries. In Section 6, we verify our algorithms by simulating the quantum process. In Section 7, we draw a conclusion.

2 PRELIMINARIES

The bit is a basic unit when we want to store data in the memory or on disks. The classical bit or the bit in classical computer usually has two *states*, which are 0 and 1. In a quantum computer, we also have the bit, which is called the *quantum bit*. For simplicity, we use *qubit* in the following. Similar to the bit in a classical computer, the qubit also has states. To describe the states of a qubit, we usually use the *Dirac notation* [10] which looks like $|\psi\rangle$. For example, $|0\rangle$ and $|1\rangle$ are two states of a qubit. This corresponds to the state 0 and state 1 of a classical bit. Different from the classical bit, the state of a qubit can be 'between' the $|0\rangle$ and $|1\rangle$. We can express this *superposition* state of a qubit with a linear combination of the two basis states:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,.$$

α and β are two complex numbers, which are usually called the *amplitudes*. Here we have $|\alpha|^2 + |\beta|^2 = 1$, where $|\alpha|^2$ means the absolute square of α. $|\psi\rangle$ can be also written as a column vector $\binom{\alpha}{\beta}$ and its *conjugate transpose* $\langle\psi|$ is $(\bar{\alpha}, \bar{\beta})$. As Dirac suggested in [1], we call the column vector a *ket* and its conjugate transpose a *bra*. Like reading a classical bit, we can measure a qubit. The measurement will destroy the superposition of the qubit so that we can only get the result state 0 with probability $|\alpha|^2$ or state 1 with probability $|\beta|^2$. For example, there is a qubit:

$$|\psi\rangle = (0.36 + 0.48i)\,|0\rangle + (0.48 - 0.64i)\,|1\rangle\,.$$

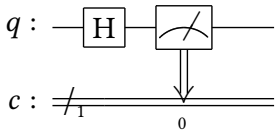
If we measure this qubit, we can get 0 with probability $|0.36 + 0.48i|^2 = 0.36^2 + 0.48^2 = 0.36$ or 1 with probability $|0.48 - 0.64i|^2 = 0.48^2 + 0.64^2 = 0.64$. Since the probability should sum to 1, $|0.36 + 0.48i|^2 + |0.48 - 0.64i|^2 = 1$ holds.

Considering there are two basis states of a qubit, if we have two qubits, the basis states will become $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. If we measure a qubit, the state of another qubit could be changed. Such a phenomenon is called *quantum entanglement* [37]. For example, assume there are two qubits q_0 and q_1:

$$\begin{aligned}|q_1q_0\rangle &= \frac{1}{\sqrt{2}}\,|0\rangle\,(0.6\,|0\rangle + 0.8\,|1\rangle) + \frac{1}{\sqrt{2}}\,|1\rangle\,(\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle)\\ &= \frac{0.6}{\sqrt{2}}\,|0\rangle + \frac{0.8}{\sqrt{2}}\,|1\rangle + \frac{1}{2}\,|2\rangle + \frac{1}{2}\,|3\rangle\end{aligned}$$

Obviously, if we measure q_1, we will get 0 or 1 with probability 0.5. However, if the result is 0, q_0 will be measured to 0 with probability $0.6^2 = 0.36$ or 1 with probability $0.8^2 = 0.64$ and if the result is 1, q_0 will be measured to 0 or 1 with probability 0.5. Similarly, we can extend this system to n qubits $q_0, q_1, \cdots q_{n-1}$ and they have 2^n basis states and corresponding 2^n amplitudes.

Similar to the classical computer, we also have wires and gates in quantum computers. The wire starting at the left hand side and ending at the right hand size is time. The gates in quantum computers are called *quantum gates*, which is an instruction at the time. For example, *Hadamard gate* [17] is a very useful quantum gate, which turns $|0\rangle$ into $(|0\rangle + |1\rangle)/\sqrt{2}$ and turns $|1\rangle$ into $(|0\rangle - |1\rangle)/\sqrt{2}$. The following shows an example of the visualization of a quantum circuit which consists of a qubit, a classical register, a Hadamard gate, and a measurement.



As above mentioned, q is the qubit. The single-line wire is a timeline, which denotes the order of the instructions coming to the qubit. The double-line wire denotes the classical register with 1 bit. The box containing H denotes the Hadamard gate. The box containing a meter denotes a measurement, which will store the result into the 0-th bit in the classical register c.

Here, a Hadamard gate can be represented by the Dirac notation:

$$|0\rangle \rightarrow (|0\rangle + |1\rangle)/\sqrt{2}; |1\rangle \rightarrow (|0\rangle - |1\rangle)/\sqrt{2}.$$

In addition, we can also use a matrix to denote the same transformation:

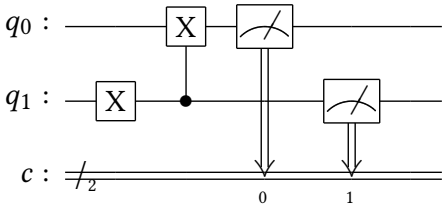
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Accordingly, the transformation of a quantum gate can be represented as a matrix multiplication. Taking the same example as above, if $|q\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$, which is a column vector $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$, we can obtain

$$H|q\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The result $(0, 1)$ means that if we measure $H|q\rangle$, we will obtain 1 with probability 1. There is a physical property that the quantum gate should be*unitary*, which means that if we write the transformation as X in the matrix representation, then $X^*X = I$, where X^* is the *conjugate transpose* of X and I is the identity matrix. To calculate the conjugate transpose of a matrix is to take the transpose of the matrix and take the *complex conjugate* $\bar{x}$ for any entry x in the transpose of the matrix. For example, it is easy to verify H is unitary, since $H^* = H$ and $H^*H = HH = I$.

Also similar to the classical computer, the gates in the quantum computer can be applied on multiple qubits. The following quantum circuit shows an example.



We have two qubits q_0 and q_1 in this example. We apply an *X-gate* on q_0, and a *controlled-X* gate on both q_0 and q_1. The X-gate is to swap the amplitudes of $|0\rangle$ and $|1\rangle$. We can represent it by the Dirac notation: $|0\rangle \rightarrow |1\rangle; |1\rangle \rightarrow |0\rangle$. Since there are two qubits, the matrix representation of this X-gate is $X \otimes I$, where X is for q_1, I is for q_0, and $\otimes$ means the calculation of tensor product [4]:

$$X \otimes I = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & 1\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ 1\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & 0\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The controlled-X gate has two parts: q_1 is the *control* qubit and q_0 is the *target* qubit. The controlled-X gate only applies an X-gate on the target qubit when the controlled qubit is $|1\rangle$, otherwise keep the target qubit unchanged. We can represent it by the Dirac notation: $|0\rangle|q_0\rangle \rightarrow |0\rangle|q_0\rangle; |1\rangle|q_0\rangle \rightarrow |1\rangle X|q\rangle_0$. In programming language, we can express this operation as: If $q_1 = 1$, then $q_0 = \text{NOT } q_0$. Its matrix presentation is

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

We list three examples.

(1) $|q_1q_0\rangle = |00\rangle$. After the X-gate on q_1, $|q_1q_0\rangle = |10\rangle$. Since $|q_1\rangle = |1\rangle$, apply an X-gate on q_0, so the result is $|11\rangle$.
(2) $|q_1q_0\rangle = |11\rangle$. After the X-gate on q_1, $|q_1q_0\rangle = |01\rangle$. Since $|q_1\rangle = |0\rangle$, do nothing, so the result is $|01\rangle$.
(3) $|q_1q_0\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. After the X-gate on q_1, $|q_1q_0\rangle = \frac{1}{\sqrt{2}}|10\rangle + \frac{1}{\sqrt{2}}|01\rangle$. The result is $\frac{1}{\sqrt{2}}|11\rangle + \frac{1}{\sqrt{2}}|01\rangle$.

It is easy to verify that the X-gate and the controlled-X gate are both unitary. In fact, any multiple qubit gate should also be unitary. Note that the inverse of a unitary matrix is also unitary, so we can know that all the quantum operations can be reversed by another quantum operation. If we can use a set of quantum gates to perform any unitary operations, we call the set a set of universal quantum gates. Usually, we describe such a quantum operation consisting of a series of quantum gates as a *quantum oracle*. A quantum oracle is also unitary, and vice versa: Each unitary matrix can be performed on qubits by a finite sequence of quantum gates. The concept of quantum oracle has been widely used in many studies such as [13, 39, 41, 44]. Since different universal gates may cause different time complexity and the general quantum computer is still at a very early stage, we usually use the query complexity to study quantum algorithms, which is to measure the number of queries to the quantum oracles.

3 RELATED WORK

Grover's algorithm is described as a database search algorithm in [13]. It solves the problem of searching a record in an unstructured list, which means that all the N records are arranged in random order. On average, the classical algorithm needs to perform $N/2$ queries to a function f which tells us if the record is the answer. More formally, for each index x, $f(x) = 1$ means the record is the answer and $f(x) = 0$ means the record is not an answer. If we have a quantum circuit to calculate this function, then we can build a Grover oracle $G : |x\rangle \rightarrow (-1)^{f(x)}|x\rangle$. Taking the advantage of quantum parallelism, Grover's algorithm can find the index of the answer with $O(\sqrt{N})$ queries to the oracle. The main idea is to first "flip" the amplitude of the answer state and then reduce the amplitudes of the other states. One such iteration will enlarge the amplitude of the answer state and $O(\sqrt{N})$ iterations should be performed until the probability that the qubits are measured to be the right answer comes close to 1. Grover mentioned in [15] that the database is supplied in the form of a quantum oracle. In fact, the Grover oracle also contains the information of the query condition, but not only a database. This oracle can recognize the solution and Grover's algorithm is from "recognizing the solution" to "knowing the solution" [33], so Grover's algorithm has limitations in database searching. For example, the generation of the Grover oracle

can be even slower than classical search [38]. However, many algorithms invoke Grover's algorithm as a subroutine due to its quadratic speedup compared to classical algorithms.

Dürr and Høyer's algorithm (DH) [11] is one of the well-known applications of Grover's algorithm and has also become a subroutine of a lot of algorithms such as [41]. DH algorithm is to find the index of the minimum record in an unsorted table. The main idea is to randomly choose a record and use Grover's algorithm to randomly choose a smaller record iteratively. The authors proved that with the total running time less than $22.5\sqrt{N} + 1.4\lg^2 N$, the algorithm can find the index of the minimum record with probability at least $\frac{1}{2}$. Compared with the quantum algorithm, the classical algorithm needs at least $\frac{N}{2}$ time. Another example was shown in [14]. To estimate the median μ of N items in an unordered list with a precision ε such that the number of records smaller than μ is less than $\frac{N}{2}(1 + |\varepsilon|)$ and the number of records greater than μ is also less than $\frac{N}{2}(1+|\varepsilon|)$, any classical algorithm needs to sample at least $\Omega(\frac{1}{\varepsilon^2})$ times. An $O(\frac{1}{|\varepsilon|})$ step quantum algorithm was proposed in this paper. As described by Grover, this method takes the same phase-shifting method as Grover's algorithm and can give an estimate of ε given μ. Combined with binary search, this algorithm can be used to find the median.

In [15], Grover et al. proposed the quantum partial search algorithm. This problem has a same condition that an unstructured list is given with a function f such that $f(x) = 1$ for a unique index x. Different from quantum search, quantum partial search only needs to find a part of the answer. If the index x is a n-bit address, then we only need to find the first k bits of x. We can regard the first k bits as a block, so we need to find the target block instead of the target record in the original problem. The authors concluded that the partial search is easier than the exact search. The best randomized partial search algorithm is expected to find the answer with $\frac{N}{2}(1 - \frac{1}{K^2})$, where $K = 2^k$. This algorithm saved $\frac{N}{2K^2}$ queries compared to the original problem. In [15], the authors proposed a better quantum algorithm that saves $\theta(\frac{1}{\sqrt{K}})$ of all queries, which is also asymptotically optimal. The main idea is to divide the original search procedure into global search and local search. We first do some iterations of global search, then do some iterations of local search in all K blocks. Note that by quantum parallelism, the local search is in parallel, so this method is a little faster than the exact search. Zhang et al. [44] discussed a harder version of the quantum partial search. In the new problem, we have multiple target records and also multiple target blocks. The target records are unevenly distributed in the list, which means that the target blocks have different numbers of target records. They solved this problem with the same main idea and the algorithm runs the fastest when target records are evenly distributed.

Wiebe et al. proposed a quantum nearest-neighbor algorithm in [41] based on DH algorithm, which shows that the quantum algorithm may provide applications to machine learning. The task is to find the closest vector to u in the training data. The training data contains M vectors $v_1, v_2, \cdots, v_M$. Then, the algorithm needs two oracles $O : |j\rangle |i\rangle |0\rangle \rightarrow |j\rangle |i\rangle \left|v_{ji}\right\rangle$ and $\mathcal{F} : |j\rangle |l\rangle \rightarrow |j\rangle |f(j,l)\rangle$, where v_{ji} is the i-th element of the v_j and $f(j,l)$ is the location of the l-th non-zero element in v_j. Using a subroutine consisting of these two quantum oracles, we can obtain $\frac{1}{\sqrt{M}} \sum_j |j\rangle (\sqrt{1 - |v_j - u|}\, |0\rangle + \sqrt{|v_j - u|}\, |1\rangle)$. Knowing that $|v_j - u|$ is the distance between u and v_j, we have all the distances encoded in the amplitudes. However, since the amplitudes can only deliver the probabilities, we cannot simply read the amplitudes. In the next step, they use amplitude estimation [6] to estimate the probabilities and store them as states, so we obtain $\frac{1}{\sqrt{M}} \sum_j |j\rangle \left||v_j - u|\right\rangle$. The final step is to use DH algorithm to find the minimum, and then we know the index of the closest vector.

There are also many studies on quantum machine learning. Li et al. [27] proposed a neural network method for conversational emotion recognition. This work did not use the quantum circuit and quantum bits but took some content from the quantum algorithm. They use a complex number vector to encode the three kinds of data and regarded the amplitudes as the probabilities of all the emotions. They proposed a quantum-like operation to update the vector iteratively. Li et al. [26] proposed a quantum-classical framework for quantum learning. In general, a machine learning framework has data, a model, a cost function, and an optimizer. Only part of their model is running on a quantum circuit. They used a quantum circuit to extract classical features and then use a fully-connected neural network to do the classification. Jerbi et al. [19] gave a quantum framework in reinforcement learning. They also used a quantum model and a classical optimizer. The quantum model is based on a parameter $\theta = (\phi, \lambda)$ where ϕ is the rotation angle and λ is the scaling parameter, and then they used sample interactions and policy gradients to update this policy parameter.

4 PROBLEM DEFINITION

This section presents the function of the quantum index. Assume that we have $N = 2^n$, where N and n are two integers. Let $[N]$ denote the set of all the integer numbers from 0 to $N - 1$. Given a set T of k d-dimensional target points in $[N]^d$ and a function $f : [N]^d \rightarrow T$, we need to build a quantum index to store all the target points and the many-to-one mapping. Using this quantum index, given a query point x, we can answer $f(x)$. In addition, by quantum parallelism, given $M = 2^m$ query points $x_1, x_2, \cdots, x_M$ in the form of $|q\rangle = \frac{1}{\sqrt{M}} \sum_i |i\rangle |x_i\rangle$, using the quantum index, we can make the following quantum transformation:

$$\frac{1}{\sqrt{M}} \sum_i |i\rangle |x_i\rangle \rightarrow \frac{1}{\sqrt{M}} \sum_i |i\rangle |f(x_i)\rangle ,$$

which means it could answer M queries simultaneously. In addition, the number of input qubits is the same as the number of output qubits, which means we do not need to use extra dn auxiliary qubits to store the answers. Since the number of qubits is still limited [36], we think it is meaningful to use fewer qubits. In our discussion, the numbers are all integers for simplicity, but they can be extended to real numbers easily. Also for simplicity, in our discussion, we assume that for each $x \in T$, we have $f(x) = x$. This constraint can be simply removed if we use a swap oracle to swap the values in T. As mentioned in Section 2, we use the query complexity to evaluate the efficiency. In other words, we assume the quantum oracles given in Section 5 cost $O(1)$ time, according to [18, 22, 24, 28, 29, 31, 32].

Notation	Definition	Remarks
N	the numbers that n qubits can represent	$N = 2^n$
n	the number of bits needed to represent one dimension	
M	the number of queries	$M = 2^m$
m	the number of bits needed to index the queries	
T	the set of target points	$T = \{T_1, \cdots, T_k\}$
k	the number of target points	
x	the input query points	
t	the number of candidate answers	$t = T_{g(x)+1} - f(x)$
$f(x)$	the many-to-one function to calculate	$f : [N]^d \to T$
$g(x)$	the index of the answer in T	$g(x) = \arg\max_j \{T_j \mid T_j \leq x\}$
ω	a primitive root of unity	$\omega = e^{-\frac{2\pi i}{t}}$
F_t	a t-by-t discrete Fourier transformation matrix	$\mathcal{F}_t = \frac{1}{\sqrt{t}}(\omega^{(j-1)(k-1)})_{1\leq j,k\leq t}$
I_t	a t-by-t identity matrix	
$\mathcal{H}$	a quantum oracle which evenly distribute the amplitudes	
$\mathcal{H}^*$	the conjugate transpose of $\mathcal{H}$	$\mathcal{H}\mathcal{H}^* = I$
$\mathcal{G}$	a quantum oracle which flip the amplitudes of the answers	
$\mathcal{O}$	a quantum oracle which flip all amplitudes except the input	
$\lvert\phi\rangle$	the state after step 1	$\lvert\phi\rangle = \mathcal{H}\lvert x\rangle$
$\lvert\psi\rangle$	the mixed state of the wrong answers	$\lvert\psi\rangle = \frac{1}{\sqrt{t-1}} \sum_{j=f(x)+1}^{T_{g(x)+1}-1} \omega^{(j-f(x))(x-f(x))} \lvert j\rangle$
θ	the angle between $\lvert\phi\rangle$ and $\lvert p\rangle$ si	
$f^*(y)$	the inverse of $f(x)$	$f^*(y) = \{x \mid f(x) = y\}$
c	the greatest number of preimages x given any $y = f(x)$	$c = \max_j \lvert f^*(T_j)\rvert$
$junk(x)$	a set of states needed in intermediate calculation process	
$C(x)$	$f*(f(x))$ followed by $c - \lvert f*(f(x))\rvert$ junks	$C(x) = \{f^*(f(x))_1, \cdots, junk(x)_1, \cdots\}$
$G(x)$	the index of x in $C(x)$	$C(x)_{G(x)} = x$

Table 1: Summary of notations

Note that f is a many-to-one function in our discussion. If f is a one-to-one function, the problem will become trivial, since we only need to use a series of swap gates which can swap the amplitudes of the states. More specifically, we only need a quantum oracle $\mathcal{A}$, where for each pair of x and y such that $f(x) = y$, we have $\mathcal{A}\lvert x\rangle = \lvert y\rangle$. We denote the conjugate transpose of $\mathcal{A}$ as $\mathcal{A}^*$ and denote the inverse of f as f^{-1}. In this case, $\mathcal{A}^*$ is obviously the quantum oracle for f^{-1}. For each point x, we can obtain $\mathcal{A}^*\mathcal{A}\lvert x\rangle = \mathcal{A}^*\lvert f(x)\rangle = \left\lvert f^{-1}(f(x))\right\rangle = \lvert x\rangle$. Since $\mathcal{A}^*\mathcal{A} = I$, we know that $\mathcal{A}$ is unitary. For example, given $N = 4$, $d = 1$ and $f(x) = (x+1) \mod 4$, we can directly obtain a design of the quantum oracle $\lvert x\rangle \to \lvert f(x)\rangle$ as $\mathcal{A} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$. If $x = 3$, we can obtain that

$$\mathcal{A}\lvert x\rangle = \mathcal{A}\lvert 3\rangle = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \lvert 0\rangle,$$

so we calculate $f(3) = 0$.

However, when it comes that f is a many-to-one function as above mentioned, things will change. Since f is not an invertible function, the quantum oracle $\mathcal{A}$ will no longer be reversible, if we build the quantum oracle by the same way. This contradicts the fact that all the quantum transformations should be unitary. Therefore, it is non-trivial to build a quantum index. Although it is difficult, we consider it very important to apply many-to-one functions in a quantum circuit, since many applications use many-to-one functions:

- Database management system. Many-to-one relationships are very common in databases. To answer a query to a many-to-one relationship is to calculate a many-to-one function.
- Classification. For example, the nearest neighbor classifier can be regarded as a many-to-one function. Given a set of k d-dimensional target points, we can divide all the points in the d-dimensional space by distances, which is a many-to-one function.
- Clustering. Given a set of clusters, we can divide all the remaining points into several groups by a criterion, which is also a many-to-one function.
- Machine learning. Dimensionality reduction is important in machine learning, which is a mapping from high-dimensional data to low-dimensional data. Since the cardinality of a higher dimensional is also greater, this mapping can also be regarded as a many-to-one function.

5 CIRCUIT-BASED DATABASE

In this section, we describe how to build a CBD and answer queries with an algorithm based on Grover search [13]. To better present the solution, we first discuss an easier case in Section 5.1. And then, we discuss the solution to the original problem in Section 5.2. In Section 5.3, we compare these two algorithms. All the notations used in this section are summarized in Table 1.

5.1 Non-Parallel 1D Index

In this section, we first discuss an easier version of the original problem. Given a set T of k target points in $[N]$ and a function $f : [N] \to T$, we need to build a quantum index to store all the target points and many-to-one mapping. Without loss of generality, we choose the lower bound function for illustration. Note that f can be an arbitrary many-to-one function. To be more specific, we have a set $T = \{T_1, T_2, \cdots, T_k\}$, where we also specially let $T_0 = 0$ and $T_{k+1} = N$ for convenience, and a function $f(x) = \max\{T_j \mid T_j \le x\}$. We use a function $g(x) = \arg\max_j\{T_j \mid T_j \le x\}$ to denote the index of the lower bound.

To store the index, we need 2 quantum oracles:

(1) $\mathcal{H} = \bigoplus_{j=0}^{k} \mathcal{F}_{(T_{j+1}-T_j)}$. $\oplus$ is direct sum. For example, $A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$. $\mathcal{F}_t$ is a t-by-t discrete Fourier transformation matrix. $\mathcal{F}_t = \frac{1}{\sqrt{t}}(\omega^{(j-1)(k-1)})_{1\le j,k\le t}$ and $\omega = e^{-\frac{2\pi i}{t}}$.
(2) $\mathcal{G} = \bigoplus_{j=0}^{k}(I_{(T_{j+1}-T_j)} - 2\,|0\rangle\langle 0|)$. I_t is a t-by-t identity matrix.

In the following, we present how to use these two oracles to calculate the answer to the lower bound query. As above mentioned, we analyze the complexity of the algorithm by the number of queries to these two quantum oracles. Now, we give a big picture of the algorithm. The algorithm mainly contains three steps. The first step is to equalize all the amplitudes of the candidate states which may contain the answer. The second step is to flip the amplitudes of the correct answer in the candidate states. The third step is to selectively rotate different states to enlarge the amplitude of the correct answer. Figure 1 shows an example of the quantum circuit when $n = 3$.

In the first step. We are given a query point x. We can easily transform $|0\rangle$ to $|x\rangle$ by applying X-gates (which are also called NOT gates) to some qubits. Then, by applying $\mathcal{H}$ to $|x\rangle$ and obtaining $|\phi\rangle$, we make all the amplitudes of candidate states equal:

$$|\phi\rangle = \mathcal{H}|x\rangle = \frac{1}{\sqrt{T_{g(x)+1} - f(x)}} \sum_{j=f(x)}^{T_{g(x)+1}-1} \omega^{(j-f(x))(x-f(x))} |j\rangle .$$

Then, we have all the amplitudes of the states in the range from $f(x)$ to $T_{g(x)+1} - 1$ equal and other states have 0 amplitudes. By the first step, we evenly distribute the probability to a subset which contains the correct answer so that we do not need to do a global search in the whole space.

In the second step, we need to use $\mathcal{G}$. If we write $\mathcal{G}$ as a matrix, we can obtain that

$$\mathcal{G}_{jk} = \begin{cases} -1 & j = k \in T; \\ 1 & j = k \notin T; \\ 0 & \text{otherwise}, \end{cases}$$

which means that $\mathcal{G}$ will multiply the amplitude of the state of the correct answer by -1 and keep other amplitudes unchanged. We use $\mathcal{G}$ to flip the amplitude of the state of the correct answer $|f(x)\rangle$ so that we can obtain

$$\mathcal{G}\mathcal{H}|x\rangle = -\frac{1}{\sqrt{T_{g(x)+1} - f(x)}} |f(x)\rangle + \frac{1}{\sqrt{T_{g(x)+1} - f(x)}} \sum_{j=f(x)+1}^{T_{g(x)+1}-1} \omega^{(j-f(x))(x-f(x))} |j\rangle .$$

For further discussion, we use a *mixed state* $|\psi\rangle$ to denote all the states of wrong answers, i.e.,

$$|\psi\rangle = \frac{1}{\sqrt{T_{g(x)+1} - f(x) - 1}} \sum_{j=f(x)+1}^{T_{g(x)+1}-1} \omega^{(j-f(x))(x-f(x))} |j\rangle ,$$

such that we can write the result as

$$\mathcal{G}\mathcal{H}|x\rangle = -\frac{1}{\sqrt{T_{g(x)+1} - f(x)}} |f(x)\rangle + \frac{\sqrt{T_{g(x)+1} - f(x) - 1}}{\sqrt{T_{g(x)+1} - f(x)}} |\psi\rangle .$$

In the third step, we need to further apply a quantum operation $(2\,|\phi\rangle\langle\phi| - I_N)$ to the qubits. Note that different from the existing two quantum oracles $\mathcal{H}$ and $\mathcal{G}$, this quantum operation should be built after the query since $|\phi\rangle$ is given by the query point x. To build it, we divide it into three sub-operations: $\mathcal{H}O\mathcal{H}^*$, where $O = 2\,|x\rangle\langle x| - I_N$. Then, we only need to build O given x:

(1) Create a $controlled - Z$ gate, where the first $k-1$ qubits are the control qubits and the last qubit is the target qubit. A Z-gate can be represented as $|1\rangle \to -|1\rangle$.
(2) For each, non-zero entry in x, create an X-gate on the left side and the right side.

For example, assume $x = 5$ and $n = 4$. We need to create a $controlled - Z$ gate. q_0, q_1 and q_2 are the control qubits. q_3 is the target qubit. x is 0101 in binary format, where the 0th and the 2rd are non-zero entries. Therefore, we add two X-gates to q_0 on the left side and the right side of the $controlled - Z$ gate and add two X-gates to q_2

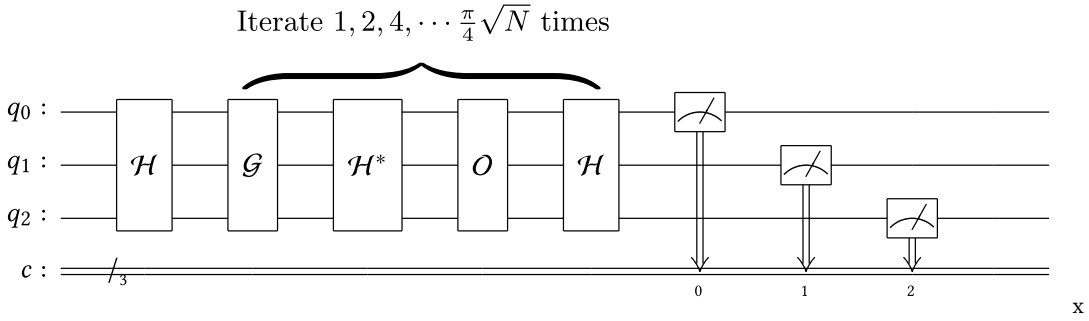


Figure 1: An example of the quantum circuit when $n = 3$

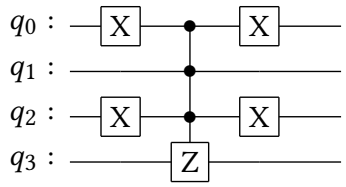


Figure 2: An example of O when $x = 5$ and $n = 4$

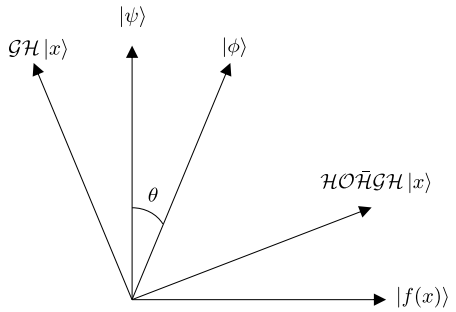


Figure 3: An illustration of the three steps

on the left side and the right side of the *controlled* $-Z$ gate. The following Figure 2 shows the quantum circuit.

The above steps build $(I_N - 2|x\rangle\langle x|)$. By flipping all the amplitudes, we can obtain $O = (2|x\rangle\langle x| - I_N)$. Let $t = T_{g(x)+1} - f(x)$. We can obtain

$$\mathcal{HOH}^* = \begin{pmatrix} \frac{2-t}{t} & \frac{2\sqrt{t-1}}{t} \\ \frac{2\sqrt{t-1}}{t} & \frac{t-2}{t} \end{pmatrix} \text{ and } \mathcal{GH}|x\rangle = \begin{pmatrix} -\frac{1}{\sqrt{t}} \\ \frac{\sqrt{t-1}}{\sqrt{t}} \end{pmatrix}.$$

And then, we can obtain

$$\mathcal{HOH}^*\mathcal{GH}|x\rangle = \frac{3t-4}{t\sqrt{t}}|f(x)\rangle + \frac{(t-4)\sqrt{t-1}}{t\sqrt{t}}|\psi\rangle,$$

which is obviously a reflection of $\mathcal{GH}|x\rangle$ across $|\phi\rangle$.

To better show the three steps, we plot the results of the three steps in a Hilbert space spanned by $|f(x)\rangle$ and $|\psi\rangle$ in Figure 3. The x-axis shows the amplitude of the state $|f(x)\rangle$ and the y-axis shows the amplitude of the state $|\psi\rangle$. By our definitions in Section 2, all the results are on the unit circle. In step one, after the initialization, all the amplitudes of the candidate states are equal. $|\phi\rangle = \mathcal{H}|x\rangle$ is at $(\frac{1}{\sqrt{t}}, \frac{t-1}{\sqrt{t}})$. We mark the angle between $|\psi\rangle$ and $|\phi\rangle$ as θ. In step two, we calculate the reflection of $|\phi\rangle$ across $|\psi\rangle$ to obtain $\mathcal{GH}|x\rangle$, which is at $(-\frac{1}{\sqrt{t}}, \frac{t-1}{\sqrt{t}})$. In step three, we further calculate the reflection of $\mathcal{GH}|x\rangle$ across $|\phi\rangle$ to obtain $\mathcal{HOH}^*\mathcal{GH}|x\rangle$, which is at $(\frac{3t-4}{t\sqrt{t}}, \frac{(t-4)\sqrt{t-1}}{t\sqrt{t}})$. By step two and step three, we enlarge the amplitude of $|f(x)\rangle$ and rotate the state by 2θ. If the amplitude of $|f(x)\rangle$ is not great enough, we can further iterate step two and step three to rotate the state until it comes close to $|f(x)\rangle$. And then, if we measure the qubits, we can obtain the correct answer with a very high probability. To analyze the number of iterations needed, we present the following Lemma 5.1.

Lemma 5.1. *The quantum operations $\mathcal{HOH}^*\mathcal{G}$ should be iterated $\frac{\pi}{4}\sqrt{t}$ times before the state comes close to $|f(x)\rangle$.*

Proof. Since step two is to calculate the reflection across $|\psi\rangle$ and step three is to calculate the reflection across $|\phi\rangle$, each iteration of step two and step three will rotate the state by 2θ. All we need to rotate by is $\frac{\pi}{2} - \theta$. Assume t is great enough. When $t \to \infty$, we can obtain $\theta = \arcsin\frac{1}{\sqrt{t}} = \frac{1}{\sqrt{t}}$ and $\frac{\pi}{2} - \theta = \frac{\pi}{2} - \frac{1}{\sqrt{t}} = \frac{\pi}{2}$. The number of iterations needed is

$$\frac{\frac{\pi}{2} - \theta}{2\theta} = \frac{\frac{\pi}{2}}{2\frac{1}{\sqrt{t}}} = \frac{\pi}{4}\sqrt{t}.$$

□

Note that the number of iterations needed depends on t. If we do more iterations after the state comes close to $|f(x)\rangle$, we will rotate the state away from $|f(x)\rangle$ and reduce the probability of obtaining the correct answer. That means we need to fix t before we know the answer. A related problem is discussed in [5], where the number of candidates is fixed but the number of answers is unknown. We propose a more straightforward solution. Create a sequence $P = \{1, 2, 4, 8, \cdots, \frac{\pi}{4}\sqrt{N}\}$, then we measure the qubits $\log\frac{\pi}{4}\sqrt{N}$ times. Before the i-th measurement, we iterate step 2 and step 3 P_i times. We answer the query with the minimum from the results of the measurements. To analyze the probability of success, we have the following Theorem 5.2.

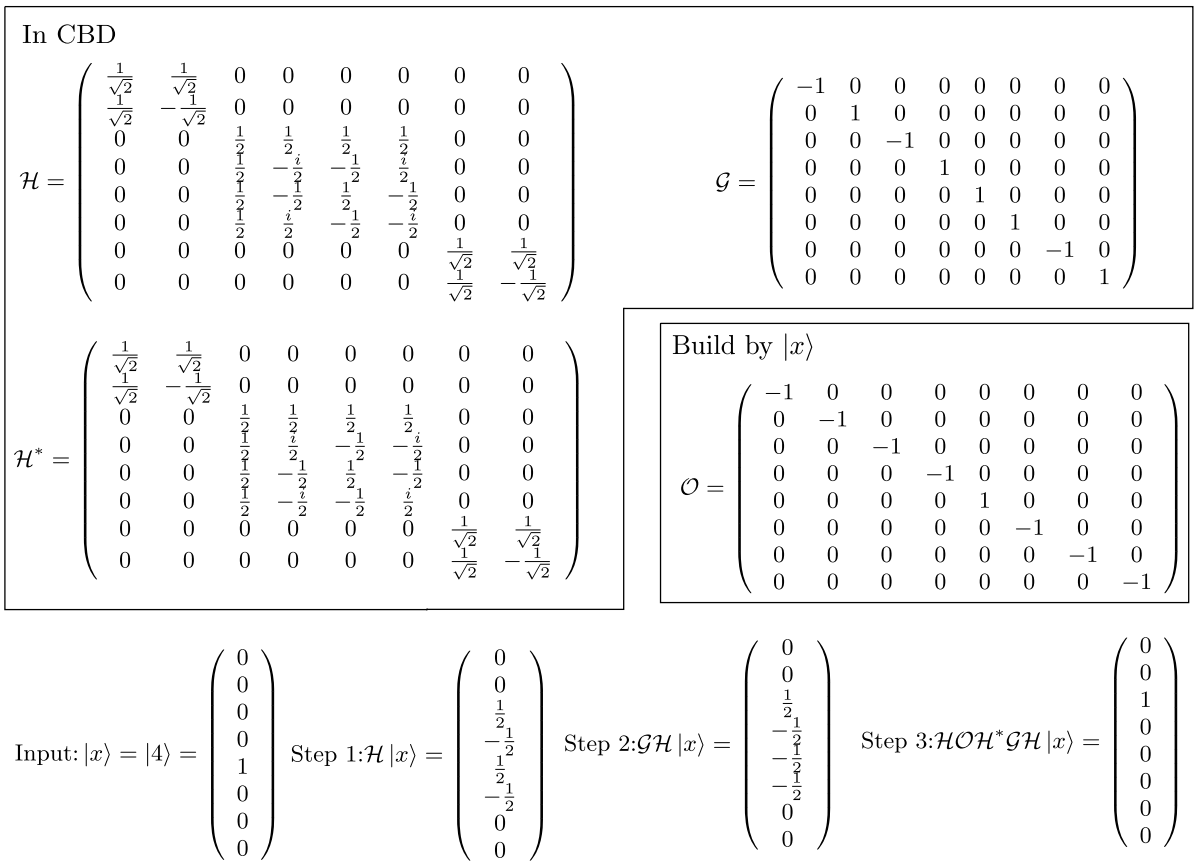


Figure 4: An illustration of the quantum process to answer $f(4)$

Theorem 5.2. *The correct answer will be obtained with probability at least* 0.8125 *with* $\log \frac{\pi}{4}\sqrt{N}$ *measurements.*

Proof. First, we can verify that it holds when t is small by exhaustion. Then assume t is great enough. By Lemma 5.1, we know that if we iterate $\frac{\pi}{4}\sqrt{t}$ times, then we rotate the state by 90 degrees, so we can obtain the maximum success probability. Obviously, if we iterate $\frac{\pi}{6}\sqrt{t} \sim \frac{\pi}{3}\sqrt{t}$ times, then rotate the state by $60 \sim 120$ degrees, so we can obtain a success probability at least $(\sin 60°)^2 = \frac{3}{4}$. Assume that there does not exist a P_i such that $\frac{\pi}{6}\sqrt{t} \le P_i \le \frac{\pi}{3}\sqrt{t}$. Then we have a P_i and a P_{i+1}, such that $P_i < \frac{\pi}{6}\sqrt{t} < \frac{\pi}{3}\sqrt{t} < P_{i+1}$. Then we have $P_{i+1} \le 2P_i < 2\frac{\pi}{6}\sqrt{t} = \frac{\pi}{3}\sqrt{t}$, which contradicts with the condition. Therefore, we must have at least one measurement will give the correct answer with probability at least $\frac{3}{4}$ in $60 \sim 120$ degrees. Similarly, we must have at least one measurement will give the correct answer with probability at least $\frac{1}{4}$ in $30 \sim 60$ degrees. Therefore, the success probability is at least $1 - (1 - \frac{3}{4})(1 - \frac{1}{4}) = 0.8125$. □

By Theorem 5.2, we know that the probability of obtaining the correct answer is at least 0.8125. To analyze the complexity of the query algorithm, we have the following Theorem 5.3.

Theorem 5.3. *The query complexity of a query to the non-parallel 1D index is* $O(\sqrt{N})$.

Proof. The total time of queries to the quantum oracles is $1 + 4\sum P_i$, which is $O(\sqrt{N})$. □

Taking the same example in Section 1, given $n = 3$, $N = 8$, a set $T = \{0, 2, 6\}$ consisting 3 1-dimensional target points in $[8]$, and a lower-bound function, we can build quantum oracles $\mathcal{H} = \mathcal{F}_2 \oplus \mathcal{F}_4 \oplus \mathcal{F}_2$ and $\mathcal{G} = (I_2 - 2|0\rangle\langle 0|) \oplus (I_4 - 2|2\rangle\langle 2|) \oplus (I_2 - 2|6\rangle\langle 6|)$ to store the data as shown in Figure 4. Then, if the input is $|x\rangle = |4\rangle$, then we can build the quantum oracle $\mathcal{O} = 2|4\rangle\langle 4| - I_8$. In step one, we use $\mathcal{H}$ to make the amplitudes of $|2\rangle$, $|3\rangle$, $|4\rangle$ and $|5\rangle$ equal. Now, all these four candidate states have probability $\frac{1}{4}$. In step two, we use $\mathcal{G}$ to flip the amplitude of $|4\rangle$ from $\frac{1}{2}$ to $-\frac{1}{2}$. In step three, we use $\mathcal{HOH}^*$ to rotate the state. Since $\theta = \arcsin\frac{1}{2} = 30°$, we rotate it by $2\theta = 60°$ to 90°, so that the probability comes to the maximum after only one iteration. We then stop the iteration and measure the qubits to get the answer.

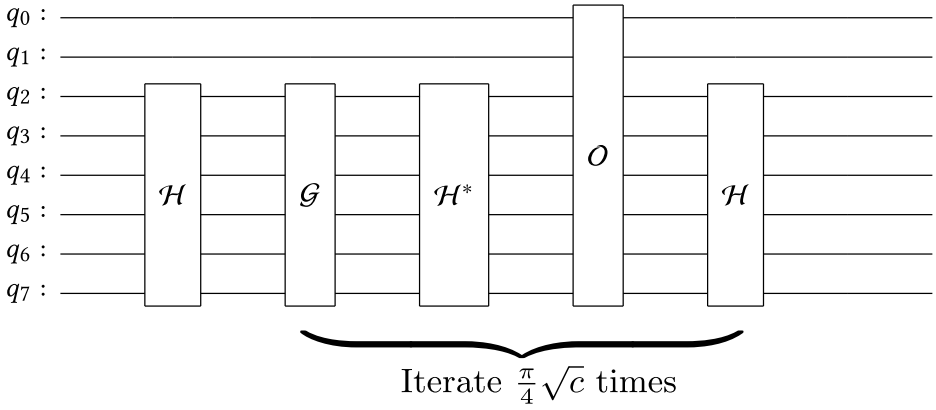


Figure 5: An example of the quantum circuit when $n = 3$, $d = 1$ and $m = 2$

5.2 Parallel Index

In this section, we discuss the original problem. Given a set T of k d-dimensional target points in $[N]^d$ and a function $f : [N]^d \to T$, we need to build a quantum index to store all the target points and many-to-one mapping. Different from Section 5.1, we need to solve high-dimensional cases and answer multiple queries in parallel. Here, the function f can be arbitrary many-to-one function. For each $y \in T$, the inverse of f is a set such that $f^*(y) = \{x \mid f(x) = y\}$. $|f^*(y)|$ is the cardinality of the set. c is the number of the candidate answers such that $c = \max_j |f^*(T_j)|$. To make all the cardinalities of the candidate answer sets for each x equal, we create a function $C(x) = \{f^*(f(x))_1, f^*(f(x))_2, \cdots, f^*(f(x))_{|f^*(f(x))|}, junk(x)_1, junk(x)_2, \cdots, junk(x)_{(c-|f^*(f(x))|)}\}$, where $junk(x)$ is some junk states for intermediate calculation process. To append $c-|f^*(f(x))|$ junks after $|f^*(f(x))|$ candidates for each x, we make all the cardinalities of candidate answer sets equal. For simplicity, we use $G(x)$ to denote the index of x in $C(x)$ such that $C(x)_{G(x)} = x$.

To store the index, we need also two quantum oracles:

(1) $\mathcal{H} : |x\rangle \left|0^{dn}\right\rangle \to \frac{1}{\sqrt{c}} \sum_{j=1}^{c} \omega^{(j-1)(G(x)-1)} \left|F(x)_j\right\rangle$, where $\omega = e^{-\frac{2\pi i}{c}}$.

(2) $\mathcal{G} : |x\rangle |y\rangle \to \begin{cases} -|x\rangle \left|0^{dn}\right\rangle & x \in T \text{ and } y = 0^{dn}; \\ |x\rangle |y\rangle & \text{otherwise.} \end{cases}$

In the following, we present how to use these two oracles to calculate the answer. If we need to answer M queries simultaneously, the queries should be given in qubits $\frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle \left|x_j\right\rangle$. M and m are two integers and $M = 2^m$. j is the index of the queries and x_j is the j-th query point. In addition, a quantum oracle O should be given with the query such that $O|j\rangle|x\rangle \to \begin{cases} -|j\rangle|x\rangle & x = x_j; \\ |j\rangle|x\rangle & x \neq x_j. \end{cases}$

We need to do the following quantum transformation:

$$\frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle \left|x_j\right\rangle \left|0^{dn}\right\rangle \to \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle \left|f(x_j)\right\rangle \left|0^{dn}\right\rangle,$$

where $\left|0^{dn}\right\rangle$ are the extra dn qubits needed in intermediate calculation process but they are not a part of the answer and can be abandoned after the calculation. Then, we have the same three steps as Section 5.1. Figure 5 shows an example of the quantum circuit when $n = 3$, $m = 2$ and $d = 1$.

In the first step. We are given multiple query points x_j in the form of $\frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle \left|x_j\right\rangle$. We need to first append dn extra qubits to obtain

$$|x\rangle = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle \left|x_j\right\rangle \left|0^{dn}\right\rangle.$$

Then, by applying $\mathcal{H}$ on the last $2dn$ qubits of $|x\rangle$ and obtaining $|\phi\rangle$, we make all the amplitudes of candidate states equal:

$$\begin{aligned} |\phi\rangle &= \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle \mathcal{H} \left|x_j\right\rangle \left|0^{dn}\right\rangle \\ &= \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle \frac{1}{\sqrt{c}} \sum_{k=1}^{c} \omega^{(k-1)(G(x_j)-1)} \left|F(x_j)_k\right\rangle. \end{aligned}$$

Then, for each query, we have all the amplitudes of the states in candidate answers equal and other states have 0 amplitudes. By the first step, we evenly distribute the probability for each query so that we do not need to search in the whole space.

In the second step, we need to use $\mathcal{G}$ to multiply the amplitude of the state of the correct answer by -1 and keep other amplitudes

unchanged. By applying $\mathcal{G}$ on the last $2dn$ qubits, we can obtain

$$\begin{aligned}
&\frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}|j\rangle\,\mathcal{G}\mathcal{H}\,|x_j\rangle\left|0^{dn}\right\rangle\\
=&\frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}|j\rangle\,\frac{1}{\sqrt{c}}\sum_{k=1}^{G(f(x_j))-1}\omega^{(k-1)(G(x_j)-1)}\,|F(x_j)_k\rangle\\
+&\frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}|j\rangle\,\frac{1}{\sqrt{c}}\sum_{k=G(f(x_j))+1}^{c}\omega^{(k-1)(G(x_j)-1)}\,|F(x_j)_k\rangle\\
-&\frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}|j\rangle\,\frac{1}{\sqrt{c}}\omega^{(G(f(x_j))-1)(G(x_j)-1)}\,|f(x_j)\rangle\left|0^{dn}\right\rangle.
\end{aligned}$$

For further discussion, we use a mixed state $|\psi_j\rangle$ to denote all the states of wrong answers for the j-th query, i.e.,

$$\begin{aligned}
|\psi_j\rangle=&\frac{1}{\sqrt{c-1}}\sum_{k=1}^{G(f(x_j))-1}\omega^{(k-1)(G(x_j)-1)}\,|F(x_j)_k\rangle\\
&+\frac{1}{\sqrt{c-1}}\sum_{k=G(f(x_j))+1}^{c}\omega^{(k-1)(G(x_j)-1)}\,|F(x_j)_k\rangle
\end{aligned}$$

such that we can write the result as

$$\frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}|j\rangle\,(\frac{1}{\sqrt{c}}\,|f(x_j)\rangle\left|0^{dn}\right\rangle+\frac{\sqrt{c-1}}{\sqrt{c}}\,|\psi_j\rangle).$$

In the third step, we need to rotate the state for the j-th query to $|f(x_j)\rangle\left|0^{dn}\right\rangle$. We first apply $\mathcal{H}^*$ on the last $2dn$ qubits. Then, we apply O on all the $m+2dn$ qubits. Last, we apply $\mathcal{H}$ on the last $2dn$ qubits. By applying these three quantum oracles, we can obtain

$$\frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}|j\rangle\,(\frac{3c-4}{c\sqrt{c}}\,|f(x_j)\rangle\left|0^{dn}\right\rangle+\frac{(c-4)\sqrt{c-1}}{c\sqrt{c}}\,|\psi_j\rangle).$$

For each query, the rotation is very similar to what is shown in Figure 3, which means we also rotate the state to $|f(x_j)\rangle\left|0^{dn}\right\rangle$ by 2θ for the j-th query. If the amplitude of the state of the correct answer is not great enough, we can further iterate step two and step three multiple times. The following Theorem 5.4 shows the number of iterations needed to obtain the maximum success probability.

Theorem 5.4. *We need to iterate $O(\sqrt{c})$ times before the state comes close to $|f(x_j)\rangle\left|0^{dn}\right\rangle$.*

Proof. Since step two is to calculate the reflection across $|\psi\rangle_j$ and step three is to calculate the reflection across $|\phi\rangle$, each iteration of step two and step three will rotate the state by 2θ. All we need to rotate by is $\frac{\pi}{2}-\theta$. We can obtain $\theta=\arcsin\frac{1}{\sqrt{c}}\le\frac{2}{\sqrt{c}}$ and $\frac{\pi}{2}-\theta=\frac{\pi}{2}-\frac{1}{\sqrt{c}}\le\frac{\pi}{2}$. The number of iterations needed is

$$\frac{\frac{\pi}{2}-\theta}{2\theta}\le\frac{\frac{\pi}{2}}{2\frac{1}{\sqrt{c}}}=\frac{\pi}{4}\sqrt{c}=O(\sqrt{c}).$$

□

By Theorem 5.4, we know that the query complexity of the algorithm is $O(\sqrt{c})$. To analyze the number of extra qubits needed, we have the following Theorem 5.5.

Theorem 5.5. *The algorithm needs at most extra nd qubits.*

Proof. In the worst case, $c=N^d$. Then for each set of candidates, we need to append N^d-1 junks, such that

$$\begin{aligned}
|\bigcup_x C(x)|&=|\bigcup_x f^*(f(x))\cup\bigcup_x junk(x)|\\
&\le|\bigcup_x f^*(f(x))|+|\bigcup_x junk(x)|\\
&\le|[N]^d|+\sum_x|junk(x)|\\
&\le N^d+N^d(N^d-1)\\
&=N^{2d}=2^{2nd}.
\end{aligned}$$

Therefore, there will be N^{2d} states and we need $2nd$ qubits totally, which contains nd input qubits. □

5.3 Comparison

In Section 5.1, we propose a non-parallel 1D algorithm for CBD. It can calculate the many-to-one function with $O(\sqrt{N})$ queries to the quantum oracles. The advantage of the algorithm is that it does not need any extra qubits and only needs the input qubits. Considering the bit width of the quantum processor may be a limit in the future, an algorithm with a lower cost should be reasonable.

In Section 5.2, we propose a parallel high-dimensional algorithm for CBD. With extra dn qubits, it can calculate the many-to-one function with $O(\sqrt{c})$ queries to the quantum oracles. Obviously, $\mathcal{G}$ is a function to check if a point is a target point, and $\mathcal{H}$ is a function similar to $C(x)$ to find out all the candidate answers. Given these two functions, any classical algorithm needs at least $O(Mc)$ queries to these functions to answer a user's query. Therefore, this quantum algorithm obtains a quadratic speedup over the classical algorithms. Furthermore, note that it can answer M queries simultaneously, but there is no M or m in the query complexity, which means that it can handle multiple queries without any extra overhead. This advantage can never be achieved by any classical algorithm.

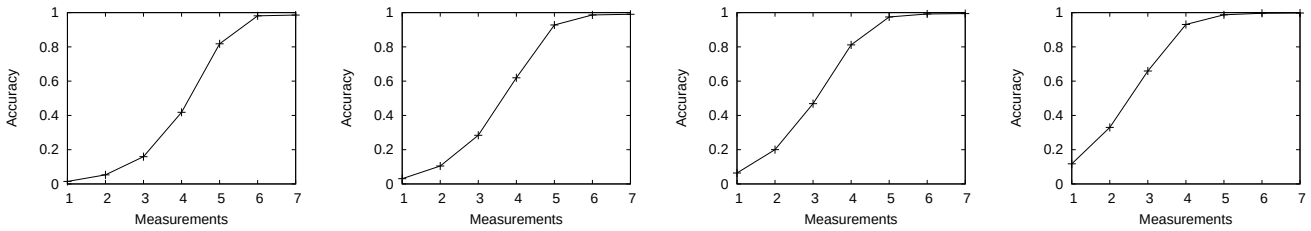
6 EXPERIMENT

The experiments were conducted on a machine running Mac OS 11.1 on Intel Core i7 with 2.6 GHz, 32 GB memory, and 512 GB SSD. We use C++ to write a program to simulate the quantum process in the algorithms. The main purpose is to verify the correctness of the two algorithms and study the accuracy of the results. We choose two datasets on Stanford Network Analysis Project (SNAP)[1], which are:

(1) **Brightkite**: A sequence of check-ins of users in a social network over the period from Apr. 2008 to Oct. 2010;
(2) **Gowalla**: A sequence of check-ins of users in a location-based social website over the period from Feb. 2009 to Oct. 2010.

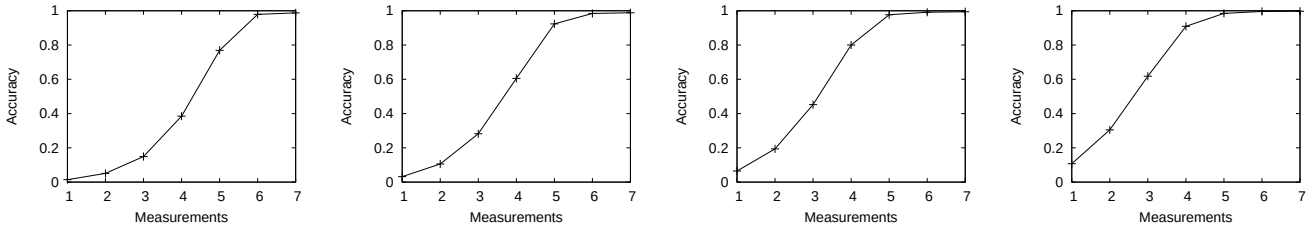
[1] https://snap.stanford.edu

, ,



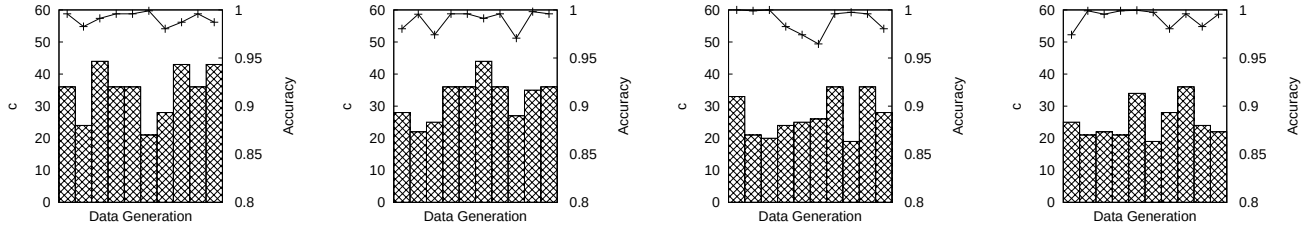
(a) $k = 8$ (b) $k = 16$ (c) $k = 32$ (d) $k = 64$

Figure 6: The first algorithm on Brightkite dataset



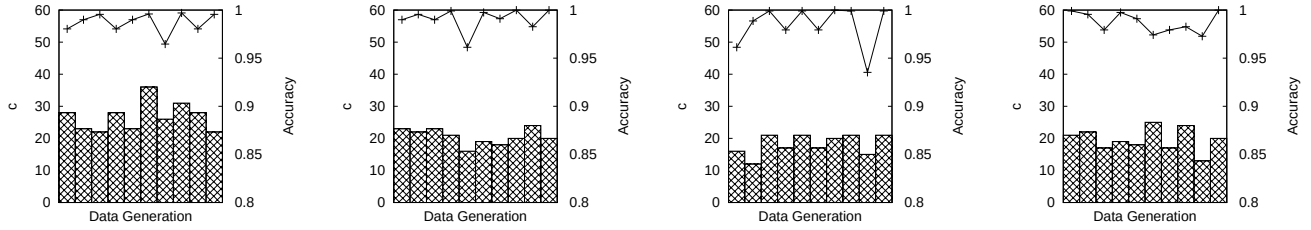
(a) $k = 8$ (b) $k = 16$ (c) $k = 32$ (d) $k = 64$

Figure 7: The first algorithm on Gowalla dataset



(a) $k = 4$ (b) $k = 6$ (c) $k = 8$ (d) $k = 10$

Figure 8: The second algorithm on Brightkite dataset



(a) $k = 4$ (b) $k = 6$ (c) $k = 8$ (d) $k = 10$

Figure 9: The second algorithm on Gowalla dataset

The data is in the format of $(time, latitude, longitude)$. Since using a classical computer to simulate a general quantum computer is very time-consuming, we limit the number of qubits to 13. Although 13 is not a great number, we need to do $(2^{13})^2$ multiplications of complex numbers to simulate one quantum operation.

Note that 13 qubits are not enough to express the whole datasets, so we need to pre-process the datasets in advance.

To simulate the first algorithm, which is a non-parallel 1D algorithm, we decide to use 12 qubits to calculate the function in $[4096]$. The many-to-one function we choose is the lower-bound function. As mentioned in Section 5.1, this many-to-one function can be arbitrary many-to-one function. Since the datasets are 3-dimensional, we choose to use $time$ attribute. We randomly choose k check-ins in the original dataset and pick the $time$ values. Then, we standardize values into $[4096]$ so that they can be expressed by 12 qubits. Then, we generate the quantum oracles $\mathcal{H}$ and $\mathcal{G}$. In a query, we randomly choose a query point x in $[4096]$ and use the first algorithm to calculate the answer $f(x)$. The first algorithm can calculate the answer with $O(\sqrt{N})$ queries to the quantum oracles, and to calculate the answer, we need to do multiple measurements with different numbers of iterations of step 2 and step 3. We record the accuracy after each measurement.

Figure 6 shows the results on Brightkite dataset and Figure 7 shows the results on Gowalla dataset. We evaluate the accuracy when $k = 8, 16, 32, 64$. For each k, we randomly generate 100 different sets of target points. For each target point, we randomly generate 10 query points. For each query point, we use the first algorithm to calculate the answer with multiple measurements. The x-axis denotes the number of measurements. The y-axis denotes the average accuracy with different target sets and different query points. Theorem 5.2 tells us that the success probability is at least 0.8125 after all the measurements. In the result, we can see all the accuracies increase rapidly with measurements. In all the experiments, the accuracies are very close to 1 after the 6-th measurement. In addition, we can find that the accuracies grow more rapidly with a greater k, which means the algorithm will perform better with a larger target set. The experiment verifies the correctness of the first algorithm. It shows that the complexity proved in Theorem 5.3 and the success probability proved in Theorem 5.2 are just theoretical guarantees, and the algorithm can perform far better than that in practice.

To simulate the second algorithm, which is a parallel d-dimensional algorithm, we decide to use 13 qubits to calculate the function. To fit the 13-qubit limit, we set $m = 1$, $n = 3$ and $d = 2$, so that the total number of qubits needed is $m+2dn = 13$. The many-to-one function we choose is the nearest-neighbor function. As mentioned in Section 5.2, this many-to-one function can be arbitrary many-to-one function. Since the datasets are 3-dimensional, we choose to use $latitude$ and $longitude$ attributes. We randomly choose k check-ins in the original dataset and pick the $(latitude, longitude)$ values. Then, we standardize values into $[8]^2$ so that they can be expressed by 6 qubits. Then, we can calculate c, which is the cardinality of the candidate answers, and expand the whole space to 12 qubits by generating the corresponding junks. Then, we generate the quantum oracles $\mathcal{H}$ and $\mathcal{G}$. In a query, we randomly choose 2 query points x_1 and x_2 in $[8]^2$, and generate a query state $|x\rangle = \frac{1}{\sqrt{2}}(|x_1\rangle + |x_2\rangle)$ and also the quantum oracle O. The second algorithm can calculate the answer with $O(\sqrt{c})$ queries to the quantum oracles. We record the accuracy for each query.

Figure 8 shows the results on Brightkite dataset and Figure 9 shows the results on Gowalla dataset. We evaluate the accuracy and c when $k = 4, 6, 8, 10$. For each k, we randomly generate 10 different sets of target points. For each target point, we randomly generate a query state. For each query state, we use the second algorithm to calculate the answer with a given quantum oracle O. The x-axis denotes the different generated target sets. The right y-axis denotes the average accuracy with different target sets. The left y-axis denotes c for these target sets. The results show that the more target points selected, the less c becomes, so that by Theorem 5.4 we need to do fewer iterations. In all the experiments, the accuracies are very close to 1, so the experiment verifies the correctness of the second algorithm.

7 CONCLUSION

To solve the problem caused by bit-based databases, we propose circuit-based databases (CBD), which use quantum circuits to store data. Unlike the bit-based database, a CBD will keep the same accuracy when answering queries. To answer queries to a given many-to-one function in the CBD, we propose two algorithms. We prove that the first quantum algorithm can answer a 1D query with $O(\sqrt{N})$ queries to the quantum oracles and it does not need any extra quantum bit. The second quantum algorithm can answer a high-dimensional query $O(\sqrt{c})$ queries to the quantum oracles. This algorithm can answer multiple queries in parallel with no extra overheads, which shows the advantage of quantum parallelism. In the experiment, we show that the two algorithms can work on real-world datasets. We consider that CBD may be a future direction for quantum databases.

REFERENCES

[1] Rudolf Bayer and Edward McCreight. 2002. Organization and maintenance of large ordered indexes. In *Software pioneers*. Springer, 245–262.
[2] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 9 (1975), 509–517.
[3] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. *Nature* 549, 7671 (2017), 195–202.
[4] Nicolas Bourbaki. 1966. *Elements of mathematics*. Hermann.
[5] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. 1998. Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics* 46, 4-5 (1998), 493–505.
[6] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. 2002. Quantum amplitude amplification and estimation. *Contemp. Math.* 305 (2002), 53–74.
[7] Bogusław Broda. 2016. Quantum search of a real unstructured database. *The European Physical Journal Plus* 131, 2 (2016), 1–4.
[8] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. 2018. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474, 2209 (2018), 20170551.
[9] Ronald De Wolf et al. 2001. *Quantum computing and communication complexity*. Citeseer.
[10] Paul Adrien Maurice Dirac. 1939. A new notation for quantum mechanics. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 35. Cambridge University Press, 416–418.
[11] Christoph Durr and Peter Hoyer. 1996. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014* (1996).
[12] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008. Quantum random access memory. *Physical review letters* 100, 16 (2008), 160501.
[13] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 212–219.
[14] Lov K Grover. 1996. A fast quantum mechanical algorithm for estimating the median. *arXiv preprint quant-ph/9607024* (1996).
[15] Lov K Grover and Jaikumar Radhakrishnan. 2005. Is partial quantum search of a database any easier?. In *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*. 186–194.

[16] Antonin Guttman. 1984. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*. 47–57.

[17] Jacques Hadamard. 1893. Resolution d'une question relative aux determinants. *Bull. des sciences math.* 2 (1893), 240–246.

[18] Akinori Hosoyamada and Yu Sasaki. 2018. Quantum Demiric-Selçuk meet-in-the-middle attacks: applications to 6-round generic Feistel constructions. In *International Conference on Security and Cryptography for Networks*. Springer, 386–403.

[19] Sofiene Jerbi, Casper Gyurik, Simon Marshall, Hans Briegel, and Vedran Dunjko. 2021. Parametrized Quantum Policies for Reinforcement Learning. *Advances in Neural Information Processing Systems* 34 (2021).

[20] Andreas Kanavos, Stavros Anastasios Iakovou, Spyros Sioutas, and Vassilis Tampakas. 2018. Large scale product recommendation of supermarket ware based on customer behaviour analysis. *Big Data and Cognitive Computing* 2, 2 (2018), 11.

[21] Ashish Kapoor, Nathan Wiebe, and Krysta Svore. 2016. Quantum perceptron models. *Advances in neural information processing systems* 29 (2016).

[22] Ruslan Kapralov, Kamil Khadiev, Joshua Mokut, Yixin Shen, and Maxim Yagafarov. 2020. Fast Classical and Quantum Algorithms for Online k-server Problem on Trees. *arXiv preprint arXiv:2008.00270* (2020).

[23] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. 2019. q-means: A quantum algorithm for unsupervised machine learning. *Advances in Neural Information Processing Systems* 32 (2019).

[24] Maria Kieferova, Ortiz Marrero Carlos, and Nathan Wiebe. 2021. Quantum Generative Training Using R\'enyi Divergences. *arXiv preprint arXiv:2106.09567* (2021).

[25] Jonas Kübler, Simon Buchholz, and Bernhard Schölkopf. 2021. The inductive bias of quantum kernels. *Advances in Neural Information Processing Systems* 34 (2021).

[26] Guangxi Li, Zhixin Song, and Xin Wang. 2021. VSQL: variational shadow quantum learning for classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 8357–8365.

[27] Qiuchi Li, Dimitris Gkoumas, Alessandro Sordoni, Jian-Yun Nie, and Massimo Melucci. 2021. Quantum-inspired neural network for conversational emotion recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 13270–13278.

[28] Tongyang Li, Shouvanik Chakrabarti, and Xiaodi Wu. 2019. Sublinear quantum algorithms for training linear and kernel-based classifiers. In *International Conference on Machine Learning*. PMLR, 3815–3824.

[29] Tongyang Li, Chunhao Wang, Shouvanik Chakrabarti, and Xiaodi Wu. 2021. Sublinear Classical and Quantum Algorithms for General Matrix Games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 8465–8473.

[30] Gui Lu Long. 2011. Duality quantum computing and duality quantum information processing. *International Journal of Theoretical Physics* 50, 4 (2011), 1305–1318.

[31] Ashley Montanaro. 2017. Quantum pattern matching fast on average. *Algorithmica* 77, 1 (2017), 16–39.

[32] María Naya-Plasencia and André Schrottenloher. 2020. Optimal Merging in Quantum k-xor and k-sum Algorithms. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 311–340.

[33] Michael A Nielsen and Isaac L Chuang. 2001. Quantum computation and quantum information. *Phys. Today* 54, 2 (2001), 60.

[34] Daniel K Park, Francesco Petruccione, and June-Koo Kevin Rhee. 2019. Circuit-based quantum random access memory for classical data. *Scientific reports* 9, 1 (2019), 1–8.

[35] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. 2014. Quantum support vector machine for big data classification. *Physical review letters* 113, 13 (2014), 130503.

[36] Marie Salm, Johanna Barzen, Frank Leymann, and Benjamin Weder. 2020. About a criterion of successfully executing a circuit in the NISQ era: what $wd \ll 1/\varepsilon_{eff}$ eff really means. In *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software*. 10–13.

[37] Erwin Schrödinger. 1935. Discussion of probability relations between separated systems. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 31. Cambridge University Press, 555–563.

[38] Raphael Seidel, Colin Kai-Uwe Becker, Sebastian Bock, Nikolay Tcholtchev, Ilie-Daniel Gheorge-Pop, and Manfred Hauswirth. 2021. Automatic Generation of Grover Quantum Oracles for Arbitrary Data Structures. *arXiv preprint arXiv:2110.07545* (2021).

[39] Peter W Shor. 1994. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 124–134.

[40] Nathan Wiebe, Daniel Braun, and Seth Lloyd. 2012. Quantum algorithm for data fitting. *Physical review letters* 109, 5 (2012), 050505.

[41] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. 2015. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Information & Computation* 15, 3-4 (2015), 316–356.

[42] William K Wootters and Wojciech H Zurek. 1982. A single quantum cannot be cloned. *Nature* 299, 5886 (1982), 802–803.

[43] Pavlo V Zahorodko, Serhiy O Semerikov, VN Soloviev, AM Striuk, MI Striuk, and Hanna M Shalatska. 2021. Comparisons of performance between quantum-enhanced and classical machine learning algorithms on the IBM Quantum Experience. In *Journal of Physics: Conference Series*, Vol. 1840. IOP Publishing, 012021.

[44] Kun Zhang and Vladimir Korepin. 2018. Quantum partial search for uneven distribution of multiple target items. *Quantum Information Processing* 17, 6 (2018), 1–20.

[45] Yao Zhang and Qiang Ni. 2020. Recent advances in quantum machine learning. *Quantum Engineering* 2, 1 (2020), e34.