 doi:10.20944/preprints202209.0358.v1

Iterative Qubits Management for Quantum Search

BY

Wenrui Mu

B.S., University of Colorado Boulder, 2020

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF SCIENCE

IN THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

AT FORDHAM UNIVERSITY

NEW YORK

May, 2022

 doi:10.20944/preprints202209.0358.v1

Contents

 doi:10.20944/preprints202209.0358.v1

 doi:10.20944/preprints202209.0358.v1

List of Tables

 doi:10.20944/preprints202209.0358.v1

List of Figures

 doi:10.20944/preprints202209.0358.v1

Chapter 1

INTRODUCTION

Before big data systems were used flexibly, most organizations could not efficiently store or manage their massive datasets due to the limited storage capacity and rigid management tools. In the past decades, developing big data applications has become increasingly important since more organizations from different industries, like healthcare systems and retailing systems, are increasingly earning benefits from the data extracted from huge datasets [1], [2]. However, with the explosion of data increases, the technical weaknesses of traditional techniques and platform efficiency have gradually become prominent, which mainly include slow responsiveness and lack of scalability, accuracy, and performance [3].

Big data-based systems have also been created and put in use since big data programming has revolutionized the world as we know it over the past decades with the explosive increase in the amount of data. In 2005, the Hadoop project was born. Hadoop was originally a project used by Yahoo to solve the problem of web search. Later, due to the efficiency of the technology, it was introduced by the Apache Software Foundation and became an open-source application [4]. Hadoop uses the Hadoop Distributed File System (HDFS) as a reliable data storage service, as well as a high-performance parallel data processing service using a technology called MapReduce, which provides a foundation that enables fast and reliable analysis of structured and complex data [5].

 doi:10.20944/preprints202209.0358.v1

The multi-faceted improvement of computer performance has provided significant advancement for computing system from various prospective, such as edge computing, mobile computing, deep learning and resource management [6]–[33]. Despite such improvements in computational power, until 2009, the learning process of deep learning models was still too slow for large-scale applications, which forced scientists to limit the size of models and training examples [34]. The turning point for this bottleneck breakthrough occurred when the training process of the neural network was ported to the GPU for training, which generated an over 35x speed-up by 2012 and made Alexnet a considerable success in 2012 Imagenet competition [35], [36]. In the training process of Alexnet in 2012, it trained for 5-6 days by using 2 GPUs [36]. Another extreme example is the machine translation architecture called "Evolved Transformer", which shows consistent improvements over the original Transformer on four well-established language tasks [37]. In the process of training this model, it cost more than 2 million training hours on GPU with millions of dollars to run [35], [37]. These examples clearly show that the cost of the researcher's model training process is increasing.

When the bottleneck of traditional computing is gradually highlighted, the advantages of computational power in other devices began to appear, including quantum computers. With recent advanced in quantum hardware, many applications have been proposed to utilize quantum computers [38]–[45].

A device performing quantum computations exploits the quantum states' properties like superposition and entanglement for calculations [46]. The concept of quantum computing was first proposed by the famous physicist Richard Feynman in 1981 [47]and is arguably the one that requires developers to make the most significant paradigm shift. In classical computing, bit is defined as the most basic unit of information [48]; similarly, "qubit" plays the same role in quantum computing [46]. Considering the fundamentals of quantum computing, a qubit can produce a coherent superposition of two logical states: $|0\rangle$ and $|1\rangle$. That is, a qubit can be in state $|0\rangle$, $|1\rangle$, or a combination of both [49]. Considering a memory of N physical bits: in classical computing, devices can only store one of 2^N possible data. If it is a quantum type, it can store 2^N data simultaneously. As N increases, its ability to keep information increases exponentially. For instance, a quantum

computer with 1024-qubit memory may store up to 2^{1024} data.

Quantum computing has expanded traditional computing based on its most essential features: quantum superposition and quantum coherence. The transformation implemented by the quantum computer for each superposition component is equivalent to a classical calculation, which is completed simultaneously and superimposed according to a certain probability amplitude to give the output result of the quantum computer [50]. This feature is called quantum parallelism [51]. To exploit quantum computers' enormous parallel processing capabilities, scientists were working to find efficient algorithms for quantum computing. In 1994, Shor discovered the first quantum algorithm, which could efficiently do factorization for a large integer N [52]. In 1997, Grover discovered another practical quantum algorithm for traversal search problems, which is suitable for solving the following problem: finding a unique element with high probability that satisfies a particular requirement from an unstructured set of N elements [53]. Thinking about the worst-case in binary tree search in the classical algorithm, where the last element traversed in a set of data is the target, the complexity is $O(N)$ [54]. Compared with that, the complexity for Grover's algorithm is only $O(\sqrt{N})$ [53], which provides a quadratic speed-up.

A simple application is to find a specific name from a phone directory with N names in a random order [53], [55]. Suppose there are 1 million names in the directory, and names will be searched one by one with an average of 500,000 searches to find the desired phone number with a $\frac{1}{2}$ chance classically [53]. In Grover's search algorithm, 1 million numbers are checked simultaneously per query. Since 1 million qubits are in a superposition state, the effect of quantum interference will cause the previous result to affect the next quantum operation. After the operation of this interference generation is repeated 1000 times, i.e., $\sqrt{N}$, the probability of obtaining the correct answer is $\frac{1}{2}$. With a few more operations, the odds of finding the desired phone number are close to 1 [53], [55].

In this paper, we propose `IQuCS`, which tackles the problem from the input size point of view. It considers a data set of *(index, value)* pairs and solely utilizes Grover's algorithm to find the targets. Based on each Grover's iteration results, `IQuCS` attempts to filter out the pairs that are not likely to be the searching targets and only send the remaining data points to the next iteration. Consequently,

the input size in each iteration is different, and the required number of qubits would be reduced. With fewer qubits iteratively, `IQuCS` provides the potential for a multi-tenant computing environment in that multiple tasks can share limited qubits. The main contributions of this paper are summarized as follows:

- We design and implement a quantum search algorithm in a hybrid system for the data set of *(index, value)* pairs. It solely relies on Grover's algorithm.
- With the reduced input size in each iteration, `IQuCS` is able to use fewer qubits to complete the search.
- We conduct both simulations with Qiskit and experiments on IBM-Q. The results demonstrate that it saves qubits consumption by up to 66.2%. Based on the results and analysis, we present lessons learned.

The remainder of this paper is structured as follows: In section 2, we introduce the related works. Section 3 presents the background of quantum computing and Grover's algorithm. In section 4, we detail to give the system design, and the design of the algorithm is provided in section 5. We provide the experimental results in Section 6. In section 7, we give a summary of our work.

 doi:10.20944/preprints202209.0358.v1

Chapter 2

RELATED WORK

Grover's algorithm, a quantum search algorithm that Lov K. Grover proposed in 1996, has been increasingly applied to the field of big data in recent years. Scientists have made great efforts to develop quantum-based applications in big data. In [56], authors introduced the implementation of data-driven tasks on the IBM Quantum Computers. Their work presented a four qubits Grover's Algorithm application and used the testing results to show the current capabilities.

Compared to the original version of Grover's Algorithm, research on improving Grover's Algorithm is also deepening. Grover's algorithm is applied to the current number of targets in the disordered quantum database, regardless of the meaning of the differences in each target. When the targets exceed half of the total items in the dataset, the algorithm will fail. In [57], to solve this problem, an improvement is proposed based on weighted indicators, in which each indicator is assigned a weight coefficient according to its weight. With these different weight coefficients, each will have equal probability with weights as quantum superposition for all target states. Then, make the phase rotations in both directions the same and determine the inner product of the two states by the superposition of the target amplitude and the system's initial state. When the inner product is $\frac{1}{2}$ or greater, the probability of finding the target will close to 100% with one Grover iteration.

For application, Grover's algorithm also has corresponding research in big data. In [58],

 doi:10.20944/preprints202209.0358.v1

authors presented a quantum algorithm based on K-nearest neighbors using Hamming distance and Grover's algorithm for the recommendation system. They analyzed the correctness of the proposed algorithm and pointed out its advantages, including exponential capacity system and response speed, independent of the classical dataset in the quantum system. Besides, in machine learning, Grover's algorithm with a dynamic selection function can be used to build a recommendation system [59]. It utilizes Hamming distance for item classification and uses Grover's Algorithm to improve the quality of recommendations. In addition, QuGAN [43] and QuClassi [38] claim to provide fabulous performance in terms of model side; however, it is obtained with only 4 dimensional on the IBM-Q platform, which is because NISQ quantum computers are low-qubits (5-7 publicly available) and noisy machines.

Variants of quantum counting have been proposed to eliminate QPE [60]–[63], a qubit-expensive operation. MLQAE [60] attempts to with multiple iterations of Grover's algorithm that combines with a maximum likelihood estimation. Wie *et al.* [61] utilizes Hadamard tests as less expensive alternatives to QPE. A simplified quantum computing algorithm that works without QPE is proposed in [62]; however, it introduces a large overhead. A recent effort, IQAE [63], can reduce the overhead through postprocessing the quantum results iteratively and only relies on Grover's operator.

`IQuCS` considers a quantum search problem of *(index, value)* pairs. From a different perspective, it focuses on reducing the input data set by filtering out non-solutions iteratively and, therefore, reducing the number of required qubits.

 doi:10.20944/preprints202209.0358.v1

Chapter 3

BACKGROUND AND MOTIVATION

3.1 Quantum State

As said, bits mathematically have states 0 and 1 in classical computing [48], and if a bit exists, its state at some point must be one between them. In physics, bit can be represented as any kind of physical system, as long as the physical system is always in one of two difference states. Computers are made up of many bits, and similarly, quantum computers are made up of quantum bits, which are also called qubits [64]. Like bits, qubits also have states, which can be represented as vectors that exist in a two-dimensional complex vector space. Theoretically, its state is represented by a column vector of length 2, and the computational basis state can be written in Eq. (1) as

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3.1}$$

In quantum mechanics, the Bloch sphere, named after physicist Felix Bloch, is a geometric representation of the space of pure states in a two-state system and is generally used when discussing qubits. Points on the sphere correspond to the pure states of the system, while the points inside it correspond to the superposition. Since $|0\rangle$ and $|1\rangle$ are the computational basis states of the qubits,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

i.e., they are a set of basis for the quantum state space, any qubit $|e\rangle$ can be expressed in the form $\alpha|0\rangle + \beta|1\rangle$, where α and β are two complex numbers whose sum of squares of the absolute value equals 1, that is, $|\alpha|^2 + |\beta|^2 = 1$ [46].

For instance, suppose

$$|q_0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix}, \tag{3.2}$$

$|q_0\rangle$ can be represented by $|0\rangle$ and $|1\rangle$ as

$$|q_0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle \tag{3.3}$$

Intuitively, qubits can be considered as a superposition of classical bit.

3.2 Qubits Representation

In quantum computing, *Statevector* is used to describe the state of a quantum system. Statevector is a vector which contains elements that representing the probability of finding a certain target in a certain condition. For instance, in classical computing, representing a target on x-axis at position 6 can be written as $x = 6$. In quantum computing, the representation is more complicated. Since each element in the statevector represents the possibility of target appearing, a target at position 6 will be represented as

$$Statevector_{example} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \tag{3.4}$$

where 1 represents the probability of target being at position 6.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

3.3 Quantum Logic Gates

A quantum gate is fundamental in the computational model of quantum computing, especially quantum circuits, which operate on a small number of qubits. Like the relationship between traditional logic gates and digital circuits, quantum logic gates is the basis of quantum circuits. Quantum gates can modify the state of qubits by taking one or more qubits to transform the state of them and get the desired output. It is worth noting that the number of inputs and outputs should be equal. That is, qubits cannot be swallowed. The quantum gates used for designing the algorithm are introduced below.

3.3.1 Single Qubit Gates

Quantum NOT gate, denoted as X gate, is a generalization of the classical NOT gate, which does the rotation around x axes of the Bloch sphere and swaps the coefficients of the two basis vectors on a single qubit.

$$NOT(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle \tag{3.5}$$

In other words, a single input-output quantum gate can be represented by a matrix of 2×2. After passing through the quantum gate, the state of a quantum gate is determined by multiplying the quantum state vector left by the value of the quantum gate matrix. The matrix corresponds to the quantum NOT gate (X gate) is [46]

$$U_X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{3.6}$$

Thus, the result of a qubit after NOT gate (X gate) conversion is

$$U_X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{3.7}$$

 doi:10.20944/preprints202209.0358.v1

Except for X-gate, single qubit gates also include Y-gate and Z-gate, which rotate around the Bloch sphere's y and z axes. Here, Y-gate converts $|0\rangle$ to $i|1\rangle$ and $|1\rangle$ to $-i|0\rangle$. Z-gate leaves the base state $|0\rangle$ unchanged and replaces $|1\rangle$ to $-|1\rangle$. Mathematically, they can be represented as a superposition of [46]

$$U_Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} U_Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.8}$$

Hadamard gate (H gate) also acts on individual qubits, which can decompose existing quantum states by coefficients [46]:

$$H(\alpha|0\rangle + \beta|1\rangle) = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle \tag{3.9}$$

It is represented by matrix as

$$U_H = \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.10}$$

and can convert $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Since each column of H-gate matrix is orthogonal, we have $HH^* = I$, where I is the identity matrix and thus H is a unitary matrix [46].

3.3.2 Multi-Qubit Gates

Computer programming is full of conditional statements; in quantum computing, we also expect qubits to interact with each other, which requires quantum gates involving two or more qubits, where one or more qubits are considered as control bits for some operation.

One of those representatives is controlled NOT gate (CNOT gate) on two qubits, where the second qubit performs a NOT operation when the first qubit is $|1\rangle$, otherwise remains unchanged. Mathematically, it can be represented by $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \theta|11\rangle$, where $|00\rangle$, $|01\rangle$, $|10\rangle$, and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

$|11\rangle$ are column vectors of length 4. This also needs to satisfy the normalization condition, that is, $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\theta|^2 = 1$. Mathematically, CNOT gate can be represented as [46]

$$CNOT(\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \theta|11\rangle) = \alpha|00\rangle + \beta|01\rangle + \gamma|11\rangle + \theta|10\rangle \tag{3.11}$$

The matrix can represent this gate:

$$U_{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.12}$$

Another gate with similar logic to CNOT gate is CCNOT gate, which also called Toffoli gate. Toffoli gate has 3 qubits as input. If the first two qubits are 1, it will invert the third qubit, otherwise all qubits remain the same [46]. Same as other quantum gates, Toffoli gate also can be expressed as a matrix.

$$U_{Toffoli} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.13}$$

SWAP gate, another two-qubit operation, based on the basis $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, can be

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

represented by the matrix as [46]:

$$U_{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

3.3.3 Application: Quantum Half-Full Adder

Different combinations of quantum gates can be used to achieve different effects, and half adder is one of them. In classical computers, a half-full adder is a logic circuit used to implement addition operations on two single binary digits, which includes two outputs: Sum (denoted as S) and Carry (denoted as C). Therefore, the sum of these two single binary digits can be expressed as $2C + S$ in decimal system [65]. We designed a simple half adder using an XOR gate to generate S and an AND gate to generate C with the following steps.

Since classical logic gates, XOR gate and AND gate are not quantum gates, we need to implement AND gate in the quantum circuit through the quantum operator and then combine them to implement the quantum half-adder. The exclusive OR gate (XOR gate) in the digit logical circuit is a gate that implements the following functions: if both inputs are false or true, the output is false; otherwise, the output is true [66].

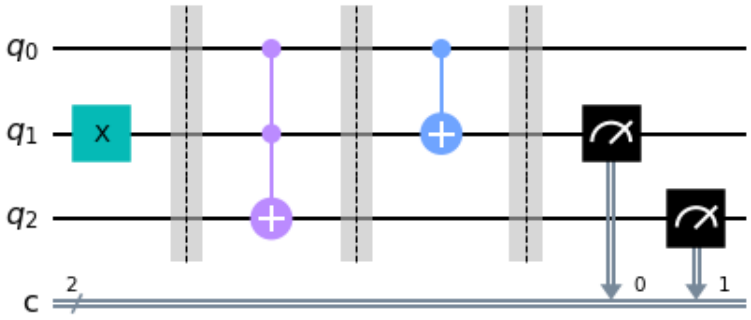


Fig. 3.1: Quantum Half-Full Adder Circuit for Input $|0\rangle$,$|1\rangle$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

The X gate changes the initial states of the quantum half-full adder. As defined, X gate can flip the quantum state [46]. Since Qiskit's default qubits are initially in the state $|0\rangle$, and the half-adder inputs require either $|0\rangle$ or $|1\rangle$, X gate is needed to do this flip. Figure 3.1 shows the circuit to do a simple quantum half-full adder circuit with inputs are $|0\rangle$ and $|1\rangle$. Above all, the X gate converts the initial state of qubit q_1 from $|0\rangle$ to $|1\rangle$.

As noted before, the CNOT gate flips the value on the second qubit depending on the state of the control bit [46]. If the value of the control bit is 1, we do the flip. Otherwise, the qubit's value remains the same. Therefore, in quantum computation, the XOR gate can be implemented directly through the CNOT gate. Considering AND operation, in this case, AND operation happens between qubits q_0 and q_1. The result of AND operation between q_0 and q_1 is XOR with qubit q_2. If the state of q_2 always is $|0\rangle$, there is no flip on the AND operation.

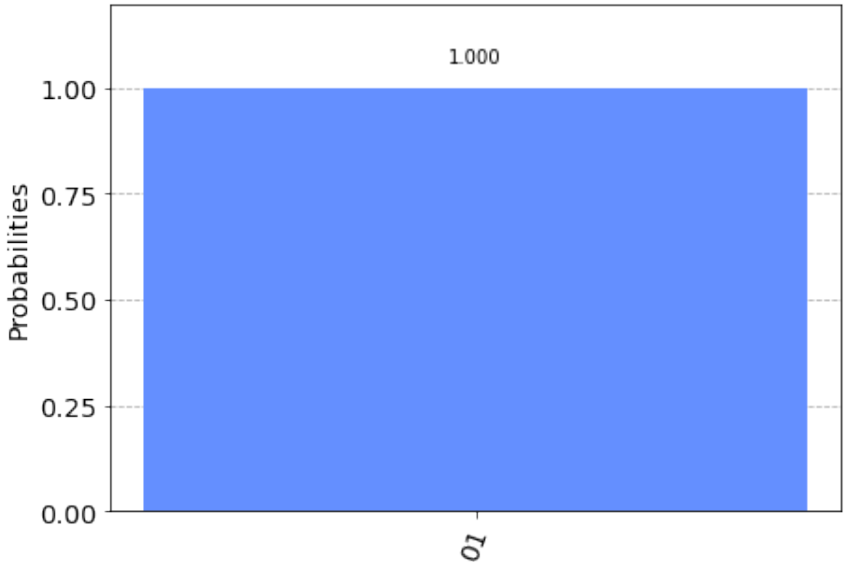


Fig. 3.2: Result of Quantum Half-Full Adder Circuit for Input $|0\rangle$,$|1\rangle$

Finally, we put these gates all into the same circuit, do the measurement, and store the result in bits c. By repeating this experiment 1024 times, Figure 3.2 shows the bar graph of the measurement results for input $|0\rangle$ and $|1\rangle$, where 0 is the carry (C), and 1 is the sum (S).

By changing the value of the input, other results of the half adder can also be calculated, which

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

Table 3.1: Quantum Half Adder Performance Results

Inputs		**Outputs**	
Input 1	Input 2	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

are listed in Table 3.1.

3.4 Quantum Entanglement

In quantum mechanics, when several particles interact with each other, the properties of each particle cannot be described individually since the properties of each particle have been integrated into the overall property. However, the properties of the overall system can be described. This phenomenon is called quantum entanglement, which only occurs in quantum systems [67]. That is, through entanglement, if the measurement determines the state of one qubit, the state of the other qubit can be determined instantaneously. This operation uses $\otimes$ to represent. With $\otimes$, the CNOT gate can be represented as:

$$U_{CNOT} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{3.15}$$

A typical example of an entangle state is the bell state [46]:

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{3.16}$$

Bell state consists of two quantum registers, which include the direct product of two $|1\rangle$ states and the direct product of two $|0\rangle$ states. If doing the measurement of the Bell state, the result can only be chosen from two. Considering probability, the probability of getting $|00\rangle$ state and $|11\rangle$ state are both 50%. Mathematically, if a quantum state $|\phi\rangle$ exists $|\phi\rangle = |a\rangle \otimes |b\rangle$, the state is defined as a separable state; otherwise, the state is considered as entangled [68]. In bell state, since it is impossible to find

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

$|a\rangle$ and $|b\rangle$ such that $|\text{Bell State}\rangle = |a\rangle \otimes |b\rangle$, it is an entangled state.

3.5 Grover's Algorithm

Grover's algorithm is a practical quantum search algorithm that quadratic speeds up an unstructured search problem, which finds the probability of each element. It has the complexity of $O(\sqrt{n})$, where N represents the number of elements in the search space, and has a quadratic speedup compared to the classical search $O(N)$ [53]. In short, the framework of Grover's algorithm is to construct an easy-to-implement operation G so that the initial state can gradually approach the target state under the iteration of the operation and then obtain the search result by measuring the target state [46].

Suppose all the data are stored in the database and the number of qubits is n, the index of each data in the database can be processed. Here, a total of 2^n data is represented, which is counted as N. Suppose there are M data will be obtained from the search. To indicate whether it is the result of our search, we create a function:

$$f(x) = \begin{cases} 1 & \text{for} \quad x = x_0 \\ 0 & \text{for} \quad x \neq x_0 \end{cases} \tag{3.17}$$

Where x_0 is the value of our search target. When we search for our target, our function value $f(x)$ is set to 1; otherwise, it is 0.

An oracle O is defined as a "black box" function that can be used as an input for another algorithm. With oracle, by considering $x = (x_0, x_1, ...x_n)$ as a binary input, oracle is defined as a classical function $f : \{0,1\}^n \to \{0,1\}^m$, which takes input with n-bit binary and generates output with m-bit binary [69]. In Grover's algorithm, a quantum oracle can identify the solution of the search problem by adding a negative phase to the solution state. That is, for every state $|x\rangle$:

$$U_\phi|x\rangle = \begin{cases} |x\rangle & \text{for} \quad x \neq \phi \\ -|x\rangle & \text{for} \quad x = \phi \end{cases} \tag{3.18}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

where U_ϕ is a diagonal matrix. In U_ϕ, the entry corresponding to the target item will have a negative phase [46].

Amplitude amplification makes the algorithm works. Start with a uniform superposition that constructed by $|s\rangle = H^{\otimes n}|0\rangle^n$ and represented as $|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$, where $N = 2^n$, the reflection angle can be calculated. Suppose vectors $|\phi\rangle$ and $|s'\rangle$ are perpendicular, the uniform superposition can be written as $|s\rangle = sin\theta|\phi\rangle + cos\theta|s'\rangle$. Thus, the reflection angle is

$$\theta = arcsin\langle s|\phi\rangle = arcsin\frac{1}{\sqrt{N}} \tag{3.19}$$

Then, we apply the oracle reflection U_f to the uniform superposition, which corresponds to a reflection of the state $|s\rangle$ about $|s'\rangle$. That is, the amplitude in front of $|\phi\rangle$ because negative. Similarly, applying another reflection U_s to the state $|s\rangle$, where $U_s = 2|s\rangle\langle s| - 1$, which maps the state to $U_s U_f|s\rangle$. After n steps, we will get the state $|\phi_n\rangle$, where $|\phi_n\rangle = (U_s U_f)^n|s\rangle$ [46].

3.5.1 Grover's Application: 4 qubits with one marked state $|1111\rangle$ from scratch

Considering the case where $N = 16$. Since $N = 2^n$ and the number of qubits is calculated by $n = log_2 N$, we need 4 qubits in this application. For the reflection angle, we have

$$\theta = arcsin\frac{1}{\sqrt{16}} = \frac{\pi}{4} \tag{3.20}$$

After n steps, we have

$$(U_s U_\phi)^n|s\rangle = sin\theta_n|\phi\rangle + cos\theta_n|s'\rangle \tag{3.21}$$

where $\theta_n = (2t+1)\theta$.

Suppose we are looking for the case $|\phi\rangle_1 = |1111\rangle$, Hadamard gates are needed to apply to the four registered qubits, which put the qubits into a superposition and make the measurement of $|0\rangle$ and

 doi:10.20944/preprints202209.0358.v1

$|1\rangle$ with equal probability.

$$U_\phi|s\rangle = U_\phi\frac{1}{\sqrt{16}}(|0000\rangle + |0001\rangle + |0010\rangle + |0011\rangle + |0100\rangle + |0101\rangle + |0110\rangle + |0111\rangle + |1000\rangle + |1001\rangle + |1010\rangle + |1011\rangle + |1100\rangle + |1101\rangle + |1110\rangle + |1111\rangle) \tag{3.22}$$

Oracle circuit will be implemented to the oracle circuit after all the qubits are initialized with the H gate, and state $|1111\rangle$ will be marked.

$$U_\phi|s\rangle = U_\phi\frac{1}{\sqrt{16}}(|0000\rangle + |0001\rangle + |0010\rangle + |0011\rangle + |0100\rangle + |0101\rangle + |0110\rangle + |0111\rangle + |1000\rangle + |1001\rangle + |1010\rangle + |1011\rangle + |1100\rangle + |1101\rangle + |1110\rangle - |1111\rangle) \tag{3.23}$$

The oracle circuit inverts the state's amplitude of $-\frac{1}{4}$ and Pauli X gates are needed to apply for certain qubits. For marked state $|1111\rangle$, oracle is created by applying a triple controlled Z gate to qubit q_3. The amplification circuit is created as follows:

- Apply H gates and Pauli X gates to all four qubits
- Apply a triple controlled Z gate to qubit q_3
- Apply H gates and Pauli X gates to all four qubits

Through these steps, the full circuit for fining marked state $|1111\rangle$ is shown in Figure 3.3.

After run the circuit in Figure 3.3 on the simulator, we got the histogram of measuring the probability of $|1111\rangle$ shown in Figure 3.4.

Considering the states, the state city graphs of the real and imaginary part of the density matrix are shown in Figure 3.5.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

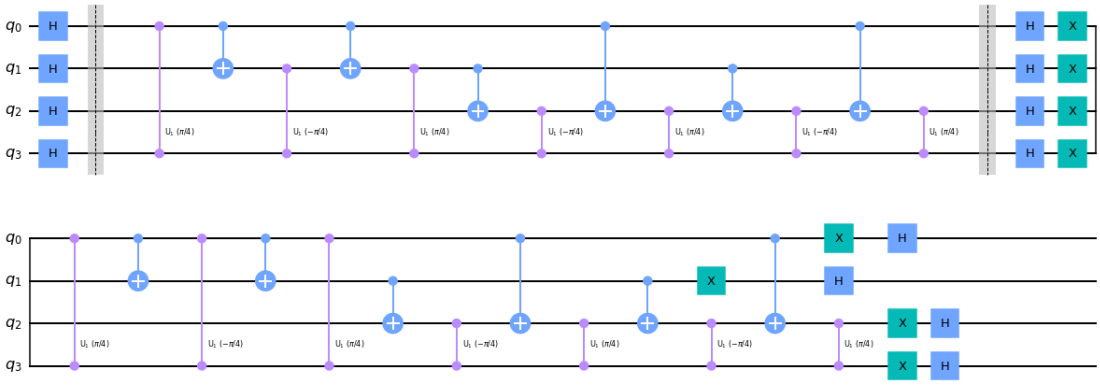


Fig. 3.3: Quantum Circuit of Measuring $|\phi\rangle = |1111\rangle$

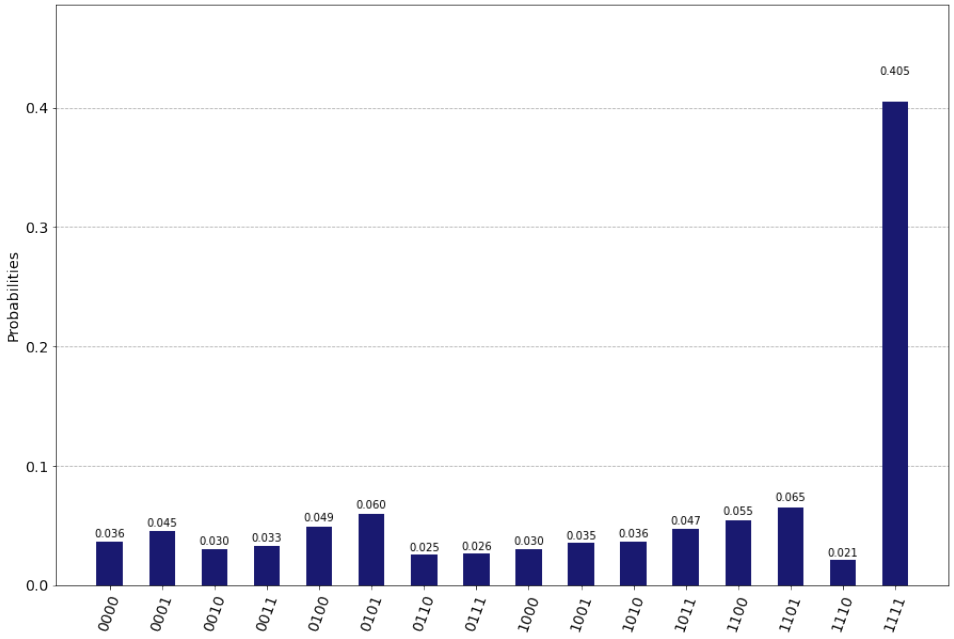


Fig. 3.4: Probability Distribution of Measuring $|\phi\rangle = |1111\rangle$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

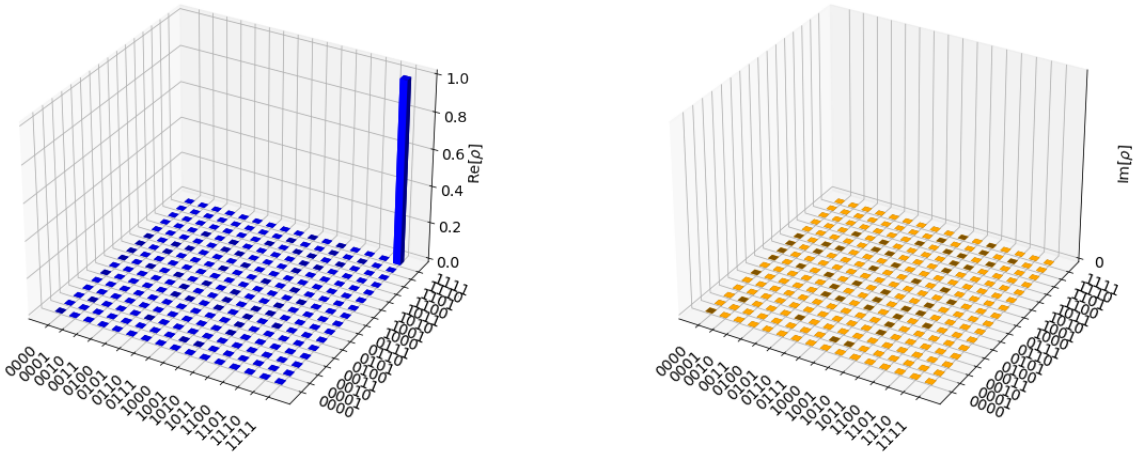


Fig. 3.5: State City of Measuring $|\phi\rangle = |1111\rangle$

3.5.2 Grover's Application: 3 Qubits with the Multiple Marked States

Grover's algorithm also can be implemented with multiple solutions in Qiskit. Considering an example when $N = 8$ with 3 marked states. Again, since $N = 2^n$ and the number of qubits needed is $n = 3$. Unstructured search with multiple solutions based on Grover's algorithm is implemented through the transformation of statevector. Data from the statevector can be a vector, circuit, or instruction. If the data is a circuit or instruction, the statevector is constructed by assuming all the qubits are initialized to the zero states. If the target value is found, we will flip the zero state in statevector into one.

Suppose the marked state is $|000\rangle$,$|101\rangle$, and $|110\rangle$, the transformation of the statevector can be represented as:

$$[0, 0, 0, 0, 0, 0, 0, 0] \rightarrow [1, 0, 0, 0, 0, 1, 1, 0] \tag{3.24}$$

The histogram for this example is shown in Figure 3.6.

States can be visualized by Figure 3.7.

The number of iterations is another factor that affects Grover's algorithm's performance. Suppose the size of the searching database is N and the number of marked states is t, the number of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

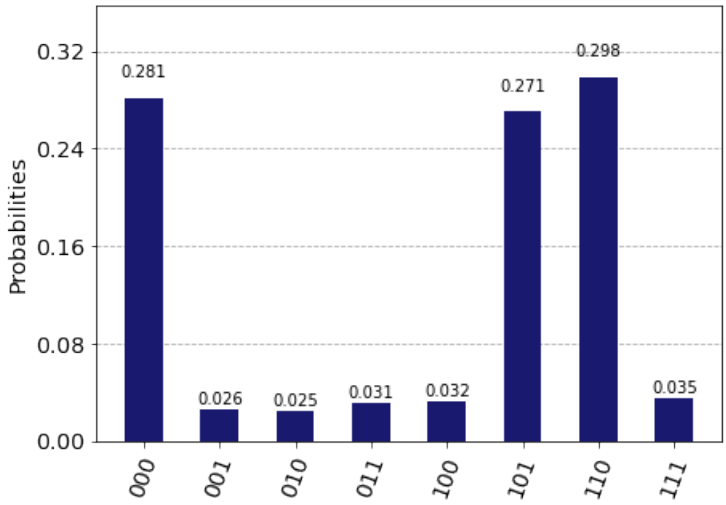


Fig. 3.6: Histogram of Measuring Multiple Marked States

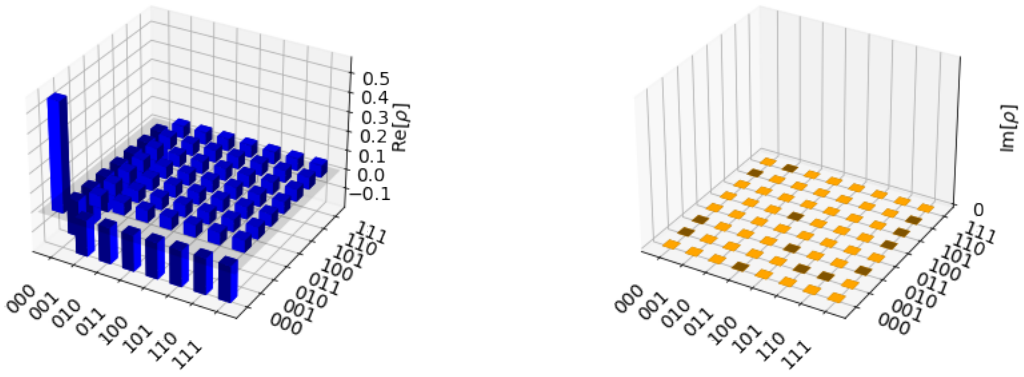


Fig. 3.7: State City of Measuring Multiple Marked States

 doi:10.20944/preprints202209.0358.v1

iterations that gives the best results is marked as $\frac{\pi}{4}(\frac{N}{t})^{\frac{1}{2}}$ [70]. However, more iterations do not mean better results for Grover's algorithm. Figure 3.8 shows the histogram for the same marked states but with different iterations. Even though the number of iterations has increased from 1 to 4, the performance of Grover's Algorithm has become worse. With more iterations, the distribution of probabilities has become more even, making it challenging to find the marked states.

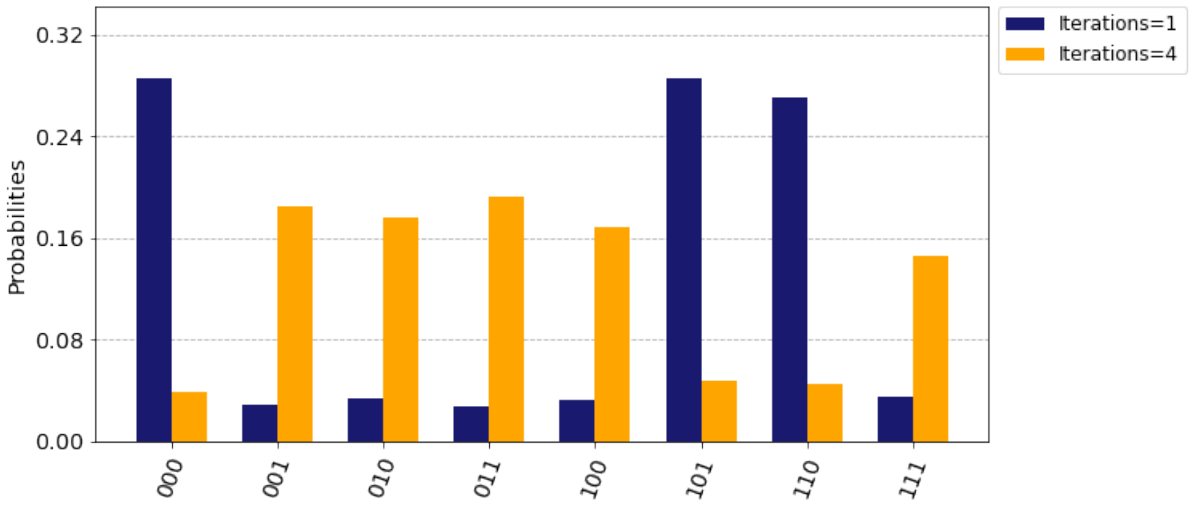


Fig. 3.8: Comparison of Different Iterations

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

Chapter 4

SYSTEM DESIGN

This chapter discusses the problem setting and system architecture of `IQuCS` that includes its framework, design logic and functionalities of key modules.

In `IQuCS`, we consider a given set of initial values as inputs, $\{v_1, ...v_i..., v_n\} \in I$. For each value, it is associated with its index. Therefore, the data set container (index, value) pairs. Specified by the users, some of the values, $\{v_i, ...v_j\} \in GS$, are defined as searching targets. The goal is to find the indexes of targets. Therefore, both indexes and values will be involved.

We utilize Grover's search algorithm to complete the task. However, the original algorithm only amplifies the amplitude of the targeted states. It fails to determine and output the targets and indexes directly. Additionally, since both indexes and values need to be encoded, the qubit requirement is larger than value-only searches. Thus, our objective is to output the targeted (index, value) pairs and reduce the number of required qubits.

Figure 4.1 illustrates the architecture of the proposed system. As a quantum-classic hybrid system, it consists of two main components, the classical computer side, and the quantum computer side. The data join the system from the classical component, where the Index Generation module is responsible for indexing the unstructured raw data (Algorithm 1). In the first iteration, the generated

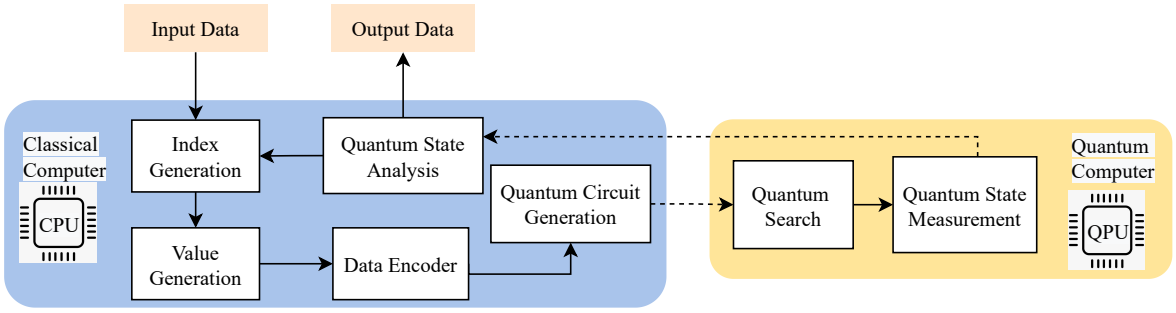


Fig. 4.1: System Architecture

indexes are designated as original indexes. Next, the data is sent to Value Generation module. It basically maps original data points to their new values in each iteration (Algorithm 2). In our design, the values update iteratively while the search goes on. Then, the *(index, value)* pairs are encoded onto qubits in the Data Encoder module, and then the quantum circuit is generated for the current iteration on the classical computer.

This circuit is passed to a quantum computer, where Grover's search algorithm is conducted with a given number of amplitude amplification. The quantum states are measured and transferred back to the classical component for further processing at the end of the search.

Upon receiving the results, Quantum State Analysis module is activated to perform Algorithm 3. If the algorithm finds all solutions, their original indexes will be returned. Otherwise, it conducts the filtering to ensure that only the potential solutions enter the next iteration. With this feature, the problem set is reduced in when Index Generation and Value Generation modules are called in the second and following rounds. Therefore, less number of qubits are required to continue the search. Meanwhile, the mappings between current indexes and original indexes and current values and original values are maintained.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

Chapter 5

SOLUTIONS

Table 5.1: Notation Table

I	Input data.
GS	The set contains searching targets.
v_i	The i^{th} values in the data set.
V_j	The input data set at iteration j.
G_i	The input [index, value] set at iteration i.
NS_i	The set that stores nonsolutions at iteration i.
PS_i	The set that stores potential solutions at iteration i.
OI	The original index function that returns indexes of V_1.
OV	The original value function that returns value of V_1.
NI	The new index function that stores latest indexes of v_i.
NV	The new value function that stores latest values of v_i.
$MpaI$	Mappings between original indexes and current indexes.
$MpaV$	Mappings between original values and current values.
R_{v_i}	The quantum state fidelity of v_i.
T_s	The filtering threshold.

5.0.1 `IQuCS` Algorithms

Searching for indexes of targeted values is a common task. Algorithm 1 assigns indexes for the given dataset iteratively and keeps mapping the original index with its current value. In the first iteration, the system calls OI, a function that stores Original Indexes, to set their initial values to -1, which indicates the not-available status (Lines 1-3). For every data point in V_1, starting from 0, it

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

Algorithm 1 Generating Indexes, GenI(V_i)

1: Inputs: $V_i, j = 0$

2: **if** $i = 1$ for V_i **then**
3: $OI(V_1) = -1$
4: **for all** $v_j \in V_1$ **do**
5: $OI(v_j) = j$
6: $MapI \leftarrow [j, j]$
7: Call GenV(v_i);
8: $j++$
9: **end for**

10: **else if** $i \neq 1$ **then**
11: **for all** $v_i \in V_i$ **do**
12: **if** $OI(v_i) \neq -1$ **then**
13: $NI(v_i) = j$
14: $MapI \leftarrow [i, j]$
15: Call GenV(v_i, j);
16: $j++$
17: **end if**
18: **end for**
19: **end if**

incrementally sets indexes, stores mapping of current indexes with its original values in $MapI$, and calls Algorithm 2 to pair each index with its corresponding data value (Lines 4-9).

Due to the data filtering process, the system requires regenerating indexes in the following iterations. In the i^{th} iteration, the algorithm neglects invalid data points that indicate by negative indexes in the previous round. For valid data points, it regenerates indexes incrementally, updates the mapping of the original index, i, with its latest value, j, in $MapI$, and assigns them to NI (Lines 10-14).

Next, it invokes $GenV$ function to pair the new index with its corresponding data value (Lines 12-18). In our system, OI always stores original indexes of filtered data points in V_1 and NI is updated iteratively to store the latest indexes. With this design, the volumn of indexes decreases as the search process continues. With fewer indexes, the required number of qubits is consequently reduced.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

Our system utilized Algorithm 2 to further reduce the qubit requirement by mapping original data values to their new values iteratively. When $GenI$ function calls it for the first time, $GenV$ pairs each value v_i in V_1 with its corresponding value j. The paired data is stored in G_1, which serves as the input for the quantum search algorithm (Lines 1-7). For the following iterations, it checks original indexes for each data point. If -1 is found, it means that this data point is marked as a nonsolution and has been filtered out in the previous round. $GenV$ ignores nonsolutions. For remaining data points with positive indexes, they are potential solutions and $GenV$ adds them to set PS_i (Lines 8-11). Next, the system searches for v_j's original value v_o by using the function OV that stores the original mapping in v_1 (Line 12). Then, a rank function is employed to generate new values based on its original value, v_o. This rank function maps the values of potential solutions to their new values in a fixed-length according to the number of elements in PS_i. The new values are stored in NV. Furthermore, $Mapv$ updates v_o's latest value to $NV(v_o)$ (Line 13-15). Finally, the new index j along with its corresponding new value ($NV(v_o)$) is inserted to G_i that serves as the input of the next iteration (Lines 16-18).

Based on the previous steps of $GenI$ and $GenV$, Algorithm 3 performs an iterative quantum search. Initially, the input dataset is sent to $GenI$, which calls $GenV$ to generate the paired input set G_1. It only happens in the first iteration, when $i = 1$ (Lines 1-2).

Next, the system invokes Grover's search with G_i as the input and GS as the searching targeted set. When i is an odd number, iteration is set to 1; otherwise, it is set to 2. This means that the Grover's operator will be invoked either 1 or 2 times depending on the value i. (Line 4-5). Please note that G_i consists of both values and indexes in their quantum states. The resulting quantum state fidelities are stored in R (Lines 6). By analyzing results, there are two scenarios.

- We first define the mean value to be the average of all possible data points in G_i, which deternines by the number of encoding qubits $\lceil \log_2 |G_i| \rceil$. When the state fidelity of v_j is lower than the mean value multiplied by a threshold, T_s, of all possible data points in G_i, which deternines by the number of encoding qubits $\lceil \log_2 |G_i| \rceil$, it indicates that v_j is unlikely to be the solution. Therefore, the algorithm adds it to the nonsolution set NS and resets its index

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

Algorithm 2 Generating Values (GenV)

1: Inputs: $V, j = 0$

2: **if** $i = 1$ for V_i **then**
3: **for all** $v_j \in V_1$ **do**
4: $OV(v_i) = v_i$
5: $MapV \leftarrow [v_i, v_i]$
6: $G_1 \leftarrow [j,\ v_i]$
7: **end for**
8: Return G_i;

9: **else if** $i \neq 1$ **then**
10: **for all** $v_j \in V_i$ **do**
11: **if** $OI(v_j) \neq -1$ **then**
12: $PS_i \leftarrow v_j$
13: $v_o = OV(v_j)$
14: $NV(v_o) = Rank(v_o)$
15: $MapV \leftarrow [v_o,\ NV(v_o)]$
16: $G_i \leftarrow [j,\ NV(v_o)]$
17: **end if**
18: Return G_i
19: **end for**
20: **end if**

value to -1. A negative index value suggests that this data point has been filtered out and will not get involved in the further iterations (Lines 7-10).

- When the state fidelity of v_j is higher than the mean value, the corresponding data point is a potential solution. In this case, it will be added to the PS_i set for further processing (Lines 11-14).

Next, potential solutions of the current iteration is compared with the previous iteration. There are two cases of comparison.

- If they are identical, it means that the algorithm has converged. Then, the current indexes of all $v_i \in PS_i$ are inserted to S. With S, the algorithm finds out original indexes of all solutions (Lines 15-19).

- When they have the difference, it suggests that the results are not stable. The algorithm will

Algorithm 3 Iterative Quantum Search

1: Initialization: $I =$ Initial Inputs
2: $G_1 =$ GenI(I)
3: i = 1

4: **while** true **do**
5: GroversSearch(G_i, GS, $(i+1) \mod 2 + 1$)
6: Update quantum state fidelities in R
7: **for** $v_j \in \{G_i\}$ **do**
8: **if** $R_{v_j} < \frac{1}{2^{\lceil \log_2 |G_i| \rceil}} \times T_s$ **then**
9: $NS_i \leftarrow v_j$
10: $OI(v_j) = -1$
11: **else**
12: $PS_i \leftarrow v_j$
13: **end if**
14: **end for**
15: **if** $PS_i = PS_{i-1}$ **then**
16: **for all** $v_i \in PS_i$ **do**
17: $S \leftarrow G_i(v_i)$
18: **end for**
19: Return $MapI(S)$
20: **else**
21: $V_i = V_i - NS_i$
22: $G_i =$ GenI(V_i)
23: $i = i + 1$
24: **end if**
25: **end while**

eliminate nonsolutions from the input. Then, it invokes $GenI$ for the next iteration on a reduced dataset (Lines 20-25).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

Chapter 6

EVALUATION

This section presents our IQuCS implementation details and results from intensive Qiskit simulations and experiments on IBM-Q.

6.1 Experimental Framework and Evaluation Metrics

We implement IQuCS with Python 3.8 and IBM Qiskit Quantum Computing simulator package. The Aer simulator is used as the backend to simulate a noise-free environment. The Grover's search module is constructed from Qiskit's amplitude amplifiers APIs. We set the number of shots to 12,000 and set threshold, $T_s = 0.85$.

The most common words in English [71] are encoded with their ranks into binaries, which act as values in our evaluation. For each data point, its initial index is the same as its value. Therefore, our workload is a data set of (key, value) pairs.

We consider two types of search scenarios, (1) The values in the data set are unique; (2) There are duplicates in the data set. Both single- and multiple-target settings are involved.

The results are compared with the original Grover's search algorithm. To presume the best

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

performance, we use *optimal_num_iterations* method to calculate the iteration number that requires knowing the number of targets beforehand. Please note that this information is NOT available to `IQuCS`. To determine the targets, it utilizes the same filter as `IQuCS`. For a specific value, if its probability is higher than the mean value (Line 8 in Algorithm 3 when $i = 1$) multiplies T_s, we assume it is a target. In the rest of this section, we use GSearch to represent this solution.

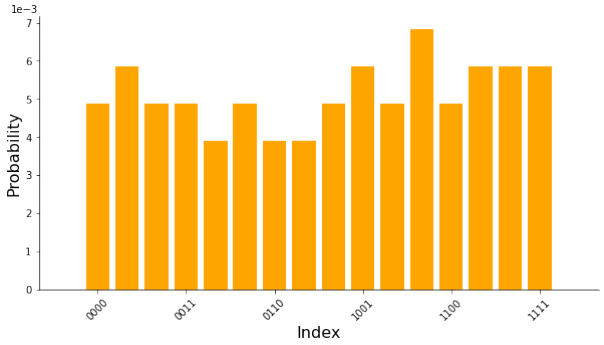
To analyze the results, we consider two metrics: (1) Accuracy; (2) Number of invocations of Grover's operator, which is called repeatedly for amplitude amplification; (3) Qubit-Virtualized Consumption (QVC); The QVC for original Grover's algorithm is straightforward since it only has one round of Grover's operator invocations. $QVC = N_q \times I$, where N_q is the number of qubits to execute the algorithm, and I is the calculated optimal number of Grover's operator invocations. The VCR is defined by the equation, $QVC = \sum_{i=1}^{i=n} C_i \times N_{q_i}$, where i is the iteration number, C_i is the number of Grover's operator invocations at iteration i and N_{q_i} is the number of qubits at iteration i.

6.2 Dataset: 10 Entries - 3 Unique Marked State

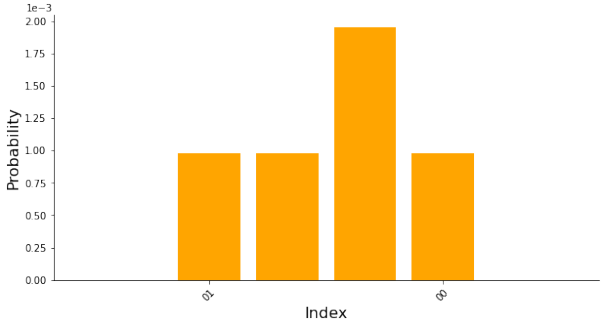
By filtering data points, iterations can be ended in three rounds. Because of redundancy, the statevector prepared for Grover's circuits is a list of zeros with a length of 2^8, and the number of qubits needed for Grover's circuit is 8. If the target was found, the statevector represents this target will flip from $|0\rangle$ to $|1\rangle$. Figure 6.1 illustrates the probability changes in three rounds. Each bar in the figure represents the probability of a key-value pair, where the highest three bars in this case represent the marked states. To get the best experimental results, we set the iterations of Grover's operator invoked as 1, 2, 1, respectively. After filtering in the first round, there are 4 data points left in the dataset. The remaining data point will be re-encoded and run in the constructed Grover's circuit again. In this case, the number of qubits has been reduced by 2. Since the marked states are found in the second round, only one more round of iteration is needed to do the verification.

As said, Grover's algorithm implements the unstructured search through amplitude amplification. Figure 6.2 illustrates the probability distribution of $invocation = 4$ and

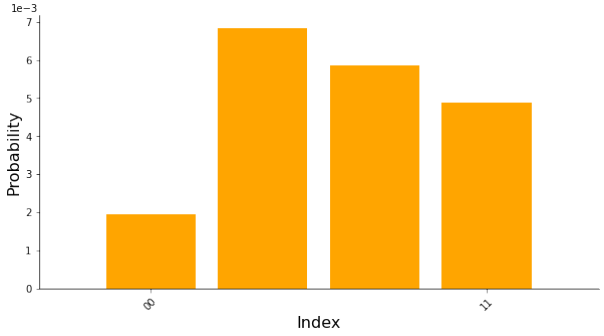
Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1



(a) Iteration 1 - Invocation 1



(b) Iteration 2 - Invocation 3

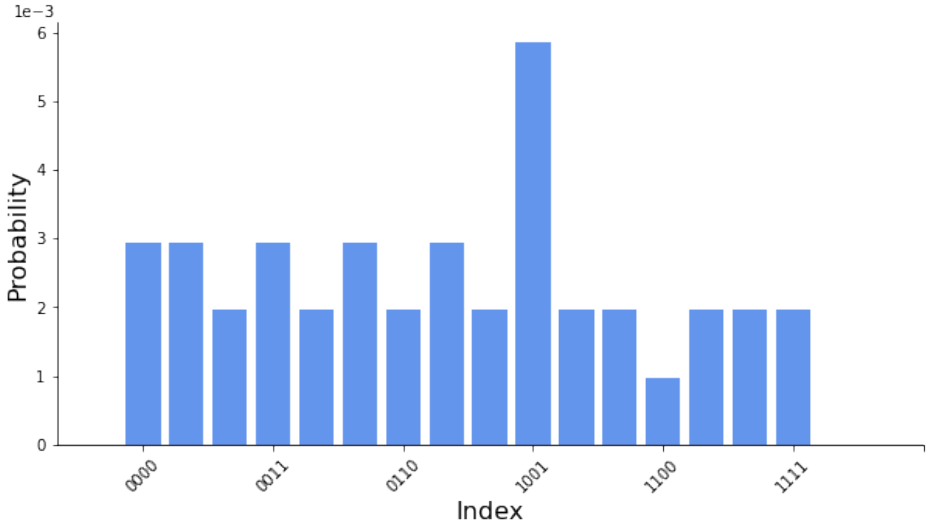


(c) Iteration 3 - Invocation 4

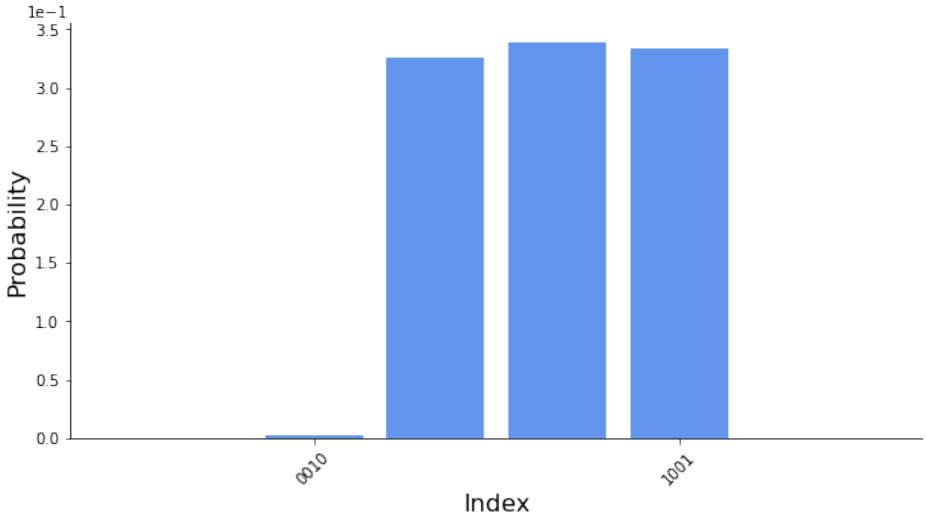
Fig. 6.1: Dataset: Length of 10 - 3 Marked States (Created)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

$invocation = 7$, which calls Grover's operator 4 and 7 times respectively. Note that the optimal invocation number is 7, which generates Figure 6.2 (b), where the target's probabilities are more than 300x times more than others. In Figure 6.1 (c), the difference is much smaller, where the non-solution with the highest probability is 2.5x times lower than the lowest one among the targets.



(a) Iteration 1 - Invocation 4



(b) Iteration 1 - Invocation 7 (Optimal)

Fig. 6.2: Dataset: Length of 10 - 3 Marked States (Original)

 doi:10.20944/preprints202209.0358.v1

6.3 Dataset: 100

We expanded the amount of data to conduct similar experiments and evaluated the algorithm. Here, the data size is 10x larger than in the previous experiment.

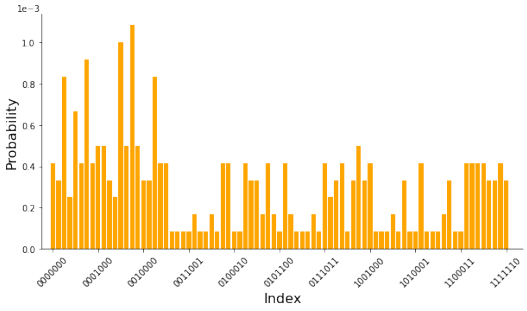
6.3.1 20 Marked States

Marked states are 20 duplicated numbers. In the first iteration, 14 qubits are needed for the key-value pair encoding, and the length of the statevector is 2^{14}. In the last iteration, the number of qubits will be reduced to 10. With a running constructed circuit through 12,000 repetitions, 16 states are found after filtering, matched with the target states. Figure 6.3 shows the probability distribution in different iterations. In this experiment, 17 data points are found, including 1 false-positive error and 4 false-negative errors.

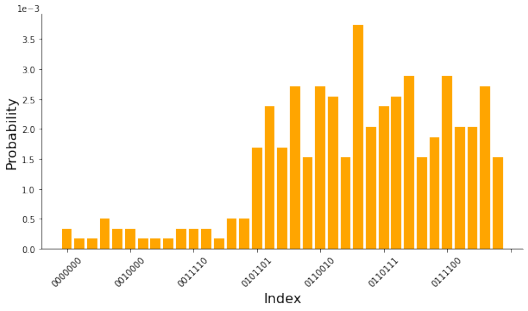
In the first iteration, the data points after filtering are 65 since the other 35 data points are excluded. By this time, the remained data contains all 20 solutions. The same situation happens in iterations 2-4, an additional 43 of the remaining data points are filtered out, all non-solutions. However, at iteration 5, another 5 data points including 3 marked states are excluded.

Figure 6.4 gives the probability distribution of invocation as 9 and 22 respestively. With the optimal invocation, $invocation = 22$, Grover's operator is called by 22 times. The number of targets found is 20, which matches the expected marked states. In terms of accuracy, our algorithm gains 97%, and the original Grover's algorithm achieves 100%. Consider QVC, GSearch's QVC is $14 \times 22 = 308$ and `IQuCS`'s QVC is $14 \times 1 + 14 \times 2 + 12 \times 1 + 10 \times 2 + 10 \times 1 + 10 \times 2 = 104$, which is a 66.2% reduction.

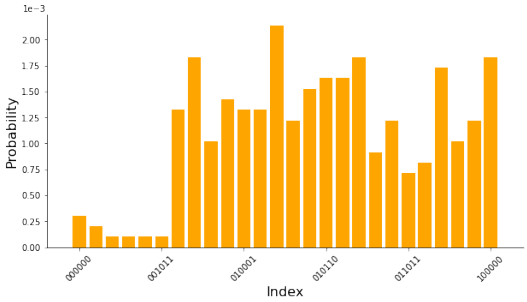
Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1



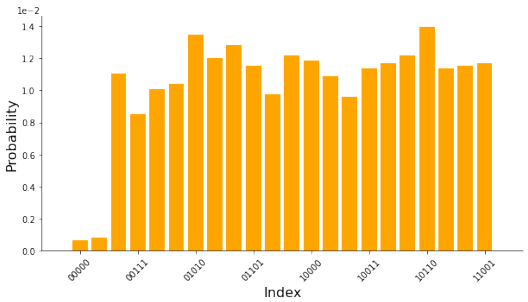
(a) Iteration 1 - Invocation 1



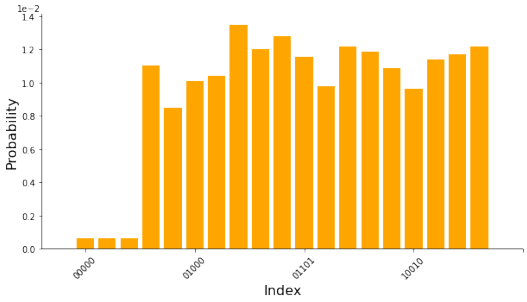
(b) Iteration 2 - Invocation 3



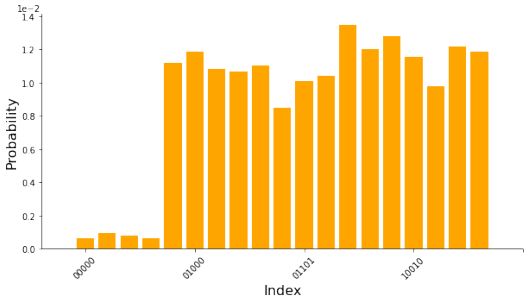
(c) Iteration 3 - Invocation 4



(d) Iteration 4 - Invocation 6



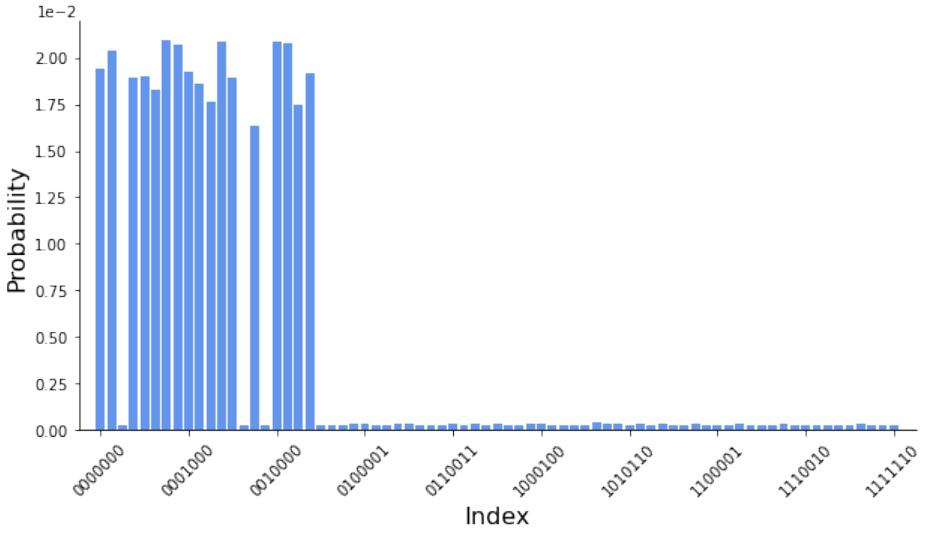
(e) Iteration 5 - Invocation 7



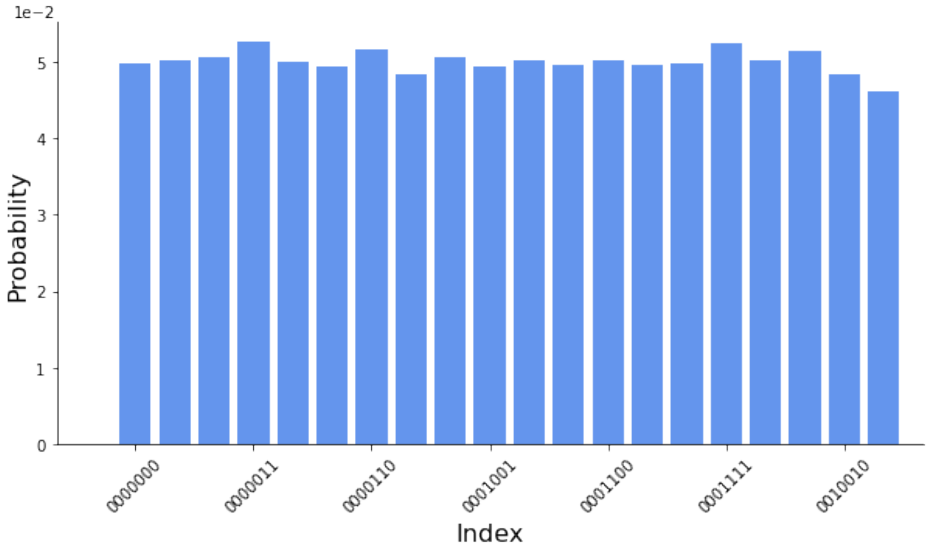
(f) Iteration 6 - Invocation 9

Fig. 6.3: Dataset: Length of 100 - 20 Duplicated Marked States (Created)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1



(a) Iteration 1 - Invocation 9



(b) Iteration 1 - Invocation 22 (Optimal)

Fig. 6.4: Dataset: Length of 100 - 20 Duplicated Marked States (Original)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

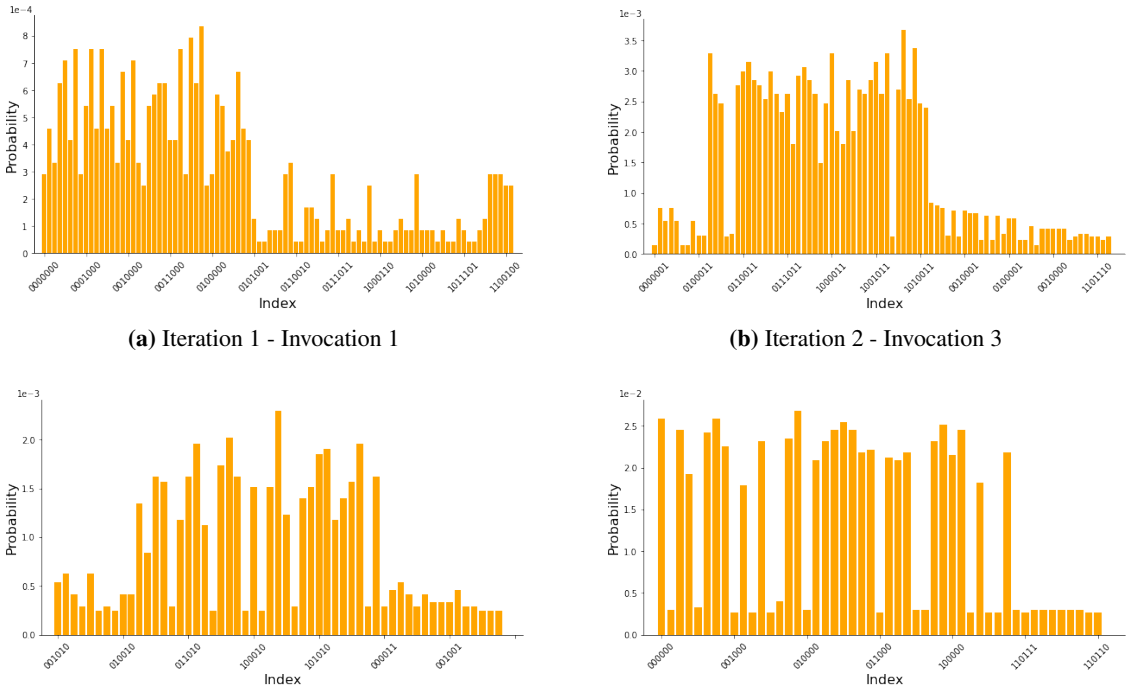
6.3.2 40 Marked States

In this section, we replaced the marked states with 40 duplicates. Again, in the first iteration, 14 qubits are required for encoding and preparing a statevector of length 2^{14}. Figure 6.5 (d) and Figure 6.6 (b) shows that both algorithms can successfully find all 40 targets. Consider the original Grover's algorithm, which requires 1 iteration, the optimal invocation number 15 makes it call Grover's operator 15 times. Comparing with our created algorithm: Since it terminates at iteration 4 and performs 6 invocations (1,2,1,2 respectively), it can reduce 60% of the invocations.

Figure 6.6 gives the iterative process of the original Grover algorithm. Figure 6.6 (a) shows the results after the 6^{th} invocation, which can allocates 41 targets. Compared with Figure 6.5, 15 non-solutions are filtered out after the first iteration, and the problem set is reduced to 85 for the encoding in the second iteration, and it is further reduced to 50 at the end of the second iteration. In the third iteration, the algorithm successfully finds all the 40 marked states. However, the system has no clues to decide the number of correct solutions. Based on Algorithm 3, the fourth iteration is performed, and the same 40 marked states are returned, which determines that these 40 marked states are all targets and the search stops.

In these experiments, both of them obtain 100% accuracy. Considering the qubits consumption, the number of qubits used will be reduced to 12 in the 4^th iteration in our algorithm. The QVC of the original Grover's search is $14 \times 15 = 210$ (it calls Grover's operator 15 times, and every time it utilizes 14 qubits, 7 qubits are used for key and value respectively). For the created algorithm, the QVC is 14×1 + 14×2 + 12×1 + 12×2=78, which gains a 62.9% reduction.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1



(a) Iteration 1 - Invocation 1

(b) Iteration 2 - Invocation 3

(c) Iteration 3 - Invocation 4

(d) Iteration 4 - Invocation 6

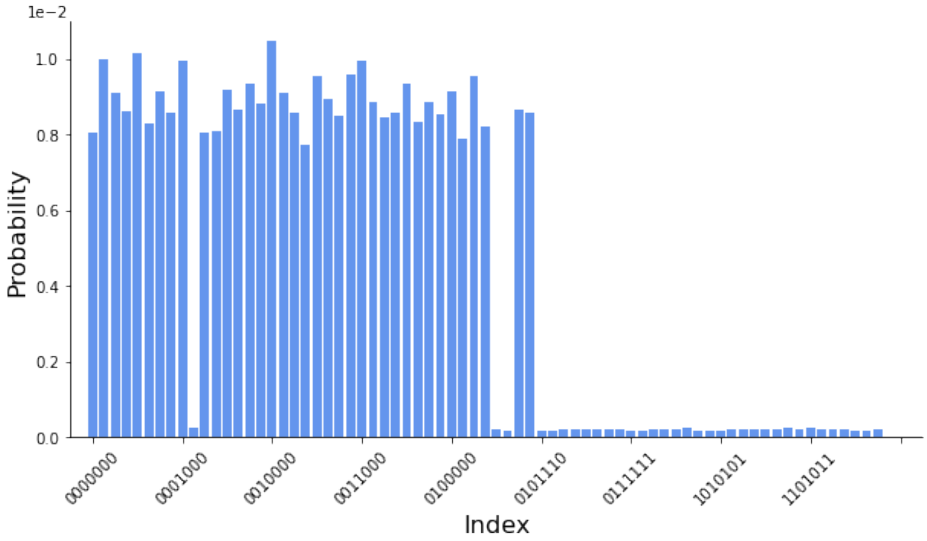
Fig. 6.5: Dataset: Length of 100 - 40 Duplicated Marked States (Created)

6.3.3 50 Marked States

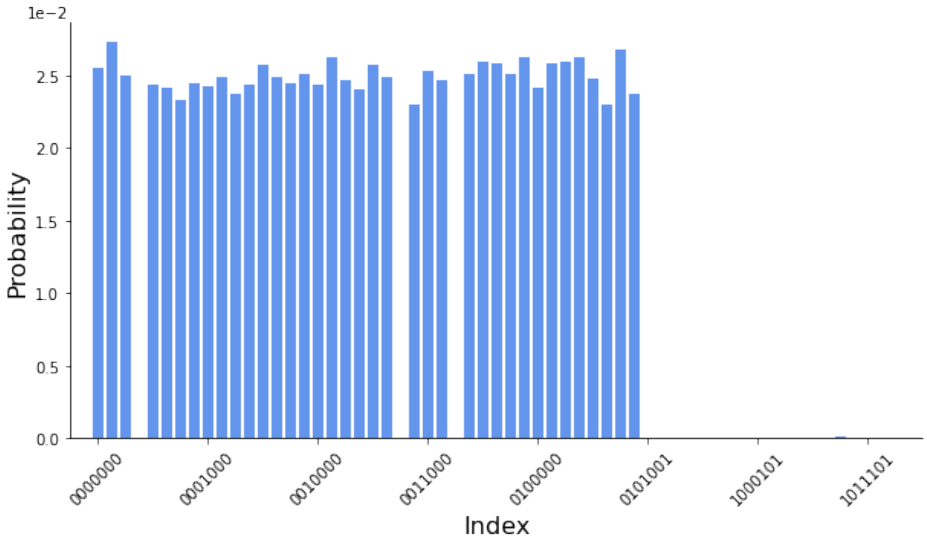
In this experiment, the marked states are set to be 50. In the first iteration, same as the 40 duplicates experiment, 14 qubits are required for encoding and preparing a statevector of length 2^{14}. Figure 6.7 (d) and Figure 6.8 (b) shows that both algorithms can successfully find all 50 marked states. Considering the original Grover's algorithm, it calls Grover's operator 15 times since its optimal invocation number is 14. Comparing with our created algorithm: Since it terminates at iteration 4 and performs 6 invocations (1,2,1,2 respectively), it is able to reduce 57.14% of the invocations.

Figure 6.8 gives the iterative process of the original Grover algorithm. Figure 6.8 (a) illustrates the results after the 6^{th} invocation, which can allocates 50 targets. Compared with Figure 6.7, 16 non-solutions are filtered out after the first iteration, and the problem set is reduced to 84 for the encoding in the second iteration, and it is further reduced to 57 at the end of the second iteration. In the third iteration, the algorithm successfully finds all the 50 marked states. Since the system is unable to

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1



(a) Iteration 1 - Invocatioin 6



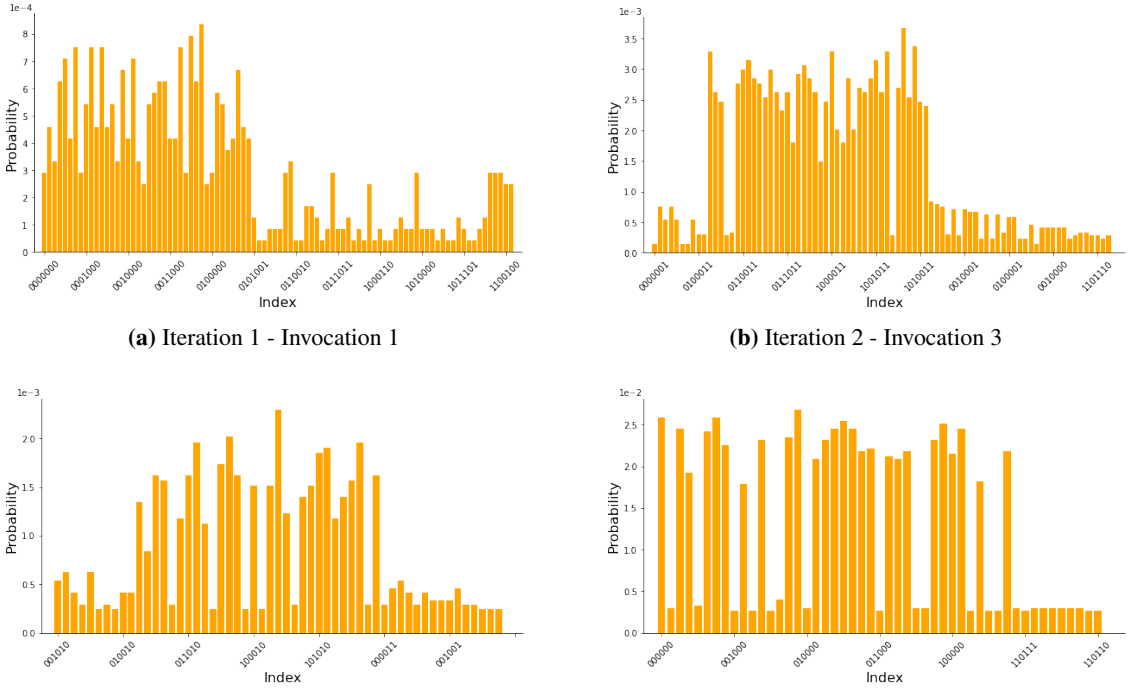
(b) Iterations 1 - Invocation 22 (Optimal)

Fig. 6.6: Dataset: Length of 100 - 40 Duplicated Marked States (Original)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

determine the targets found are marked states, we performed the fourth iteration. Since targets found after the fourth iteration are exactly the same as the third iteration, we said these 50 data points found are the marked states and the loop terminated.

To evaluate the algorithm based on these experiments, both obtain 100% accuracy. Considering the qubits consumption, the number of qubits used will be reduced to 12 in the 4^th iteration in our algorithm. The QVC of the original Grover's search is $14 \times 14 = 196$ since it calls Grover's operator 14 times and every time it utilizes 14 qubits, 7 qubits are used for key and value respectively. For the created algorithm, the QVC is 14×1 + 14×2 + 12×1 + 12×2=78, which gains a 60.2% reduction.



(a) Iteration 1 - Invocation 1

(b) Iteration 2 - Invocation 3

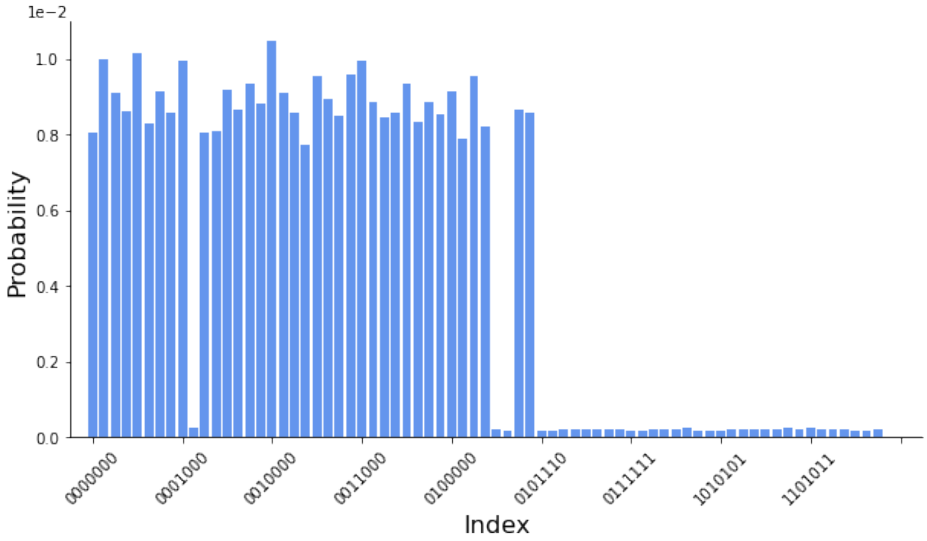
(c) Iteration 3 - Invocation 4

(d) Iteration 4 - Invocation 6

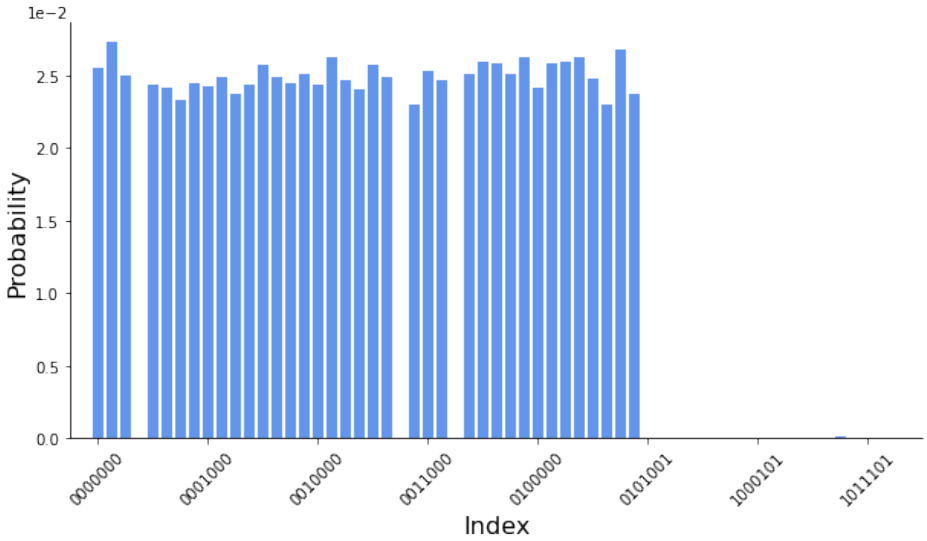
Fig. 6.7: Dataset: Length of 100 - 50 Duplicated Marked States (Created)

Figure 6.9 presents the `IQuCS` qubits consumption for three sets of experiments, in relative to GSearch, of each invocation. We assume the consumption of GSearch is 1. At the first Grover's operator invocation, the consumption is always the same of both `IQuCS` and GSearch since they have the same initial input size. GSearch's input set remains as the algorithms proceed, and only the probability of the individual data point updates after each call. The input set reduces with

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1



(a) Iteration 1 - Invocation 6



(b) Iterations 1 - Invocation 15 (Optimal)

Fig. 6.8: Dataset: Length of 100 - 50 Duplicated Marked States (Original)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 September 2022 doi:10.20944/preprints202209.0358.v1

Table 6.1: Experiments on IBM-Q

Machines	GSearch	IQuCS
Belem	4.38/4.43/4.92	4.31/9.11/13.51
Lima	6.28/6.73/7.15	6.36/12.62/19.21
Quito	4.21/4.56/4.75	4.47/8.42/13.04
Jakarta	5.85/6.14/6.16	6.01/11.72/16.98

IQuCS since it filters out data points iteratively. Therefore, it may require fewer qubits in the next iteration. The shadowed spaces on the figure show the saved qubit resources of IQuCS.

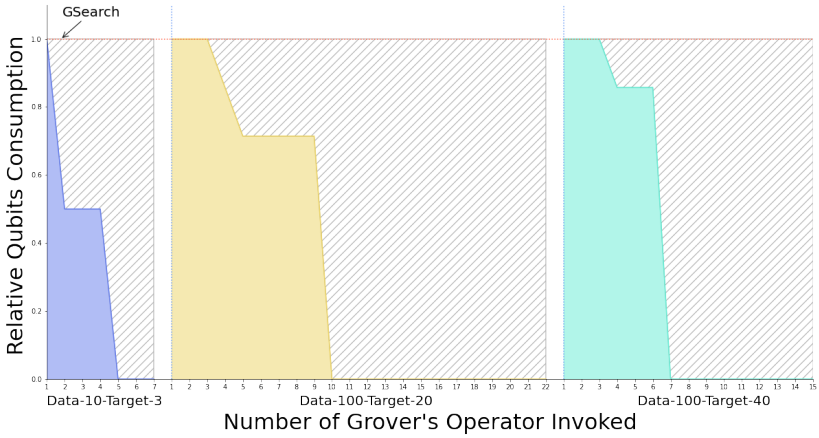


Fig. 6.9: Qubits Consumption Comparison

6.4 IBM-Q Experiments

We conduct the experiments on IBM-Q quantum computers with 5-qubits, Belem, Lima, and Quito, and 7-qubits Jakarta. The value-only data is considered due to limited qubits. In these experiments, we focus on the execution time and set the number of invocations to 1,3, and 5, the number of targets to 1. Table 6.1 presents the results in seconds. When invocation is 1, they perform similarly since both of them have only 1 iteration. GSearch's time cost is stable and IQuCS grows. The reason is that IQuCS requires multiple queries to the quantum computer, which has to compile and initialize the circuits for each query that generates significant overhead. While the total number of invocations reduces, the saved time cost fails to overcome the loss of multiple initialization phases.

 doi:10.20944/preprints202209.0358.v1

Chapter 7

CONCLUSION

This project studies a quantum index search problem within a quantum-classical system. Based on the original Grover's algorithm, we propose `IQuCS` that queries quantum computer iteratively and process the quantum results on the classical part. With the assistance of classical computers, `IQuCS` can reduce the problem set for each query. Due to this reduction, `IQuCS` requires fewer qubits. Through the iterative management, `IQuCS` achieves a reduction of qubit visualized consumption, up to 66.2%, with reasonable accuracy. However, depending on the threshold value, `IQuCS` may suffer from true negative scenarios, where targets are filtered out without a recovery mechanism.

Our work provides a general step forward in quantum resource management for the future hybrid quantum cloud era. With the improved consumption, the qubits can be shared with other users. There is, however, still significant progress to be made in this domain. Depending on the threshold value, `IQuCS` may suffer from true negative scenarios, where targets are filtered out without a recovery mechanism. In addition, the initialization of each quantum iteration is an expensive operation in the current NISQ era. Efficient collaboration and task distribution between quantum and classical computers in a hybrid cluster should be investigated.

 doi:10.20944/preprints202209.0358.v1

Bibliography

[1] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile networks and applications*, vol. 19, no. 2, pp. 171–209, 2014.

[2] B. Schmarzo, *Big Data: Understanding how data powers big business*. John Wiley & Sons, 2013.

[3] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Big data technologies: A survey," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 4, pp. 431–448, 2018.

[4] A. C. Murthy and D. Eadline, *Apache Hadoop YARN: moving beyond MapReduce and batch processing with Apache Hadoop 2*. Pearson Education, 2014.

[5] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, 2013, pp. 1–16.

[6] Y. Mao, V. Sharma, W. Zheng, L. Cheng, Q. Guan, and A. Li, "Elastic resource management for deep learning applications in a container cluster," *IEEE Transactions on Cloud Computing*, 2022.

[7] Y. Mao, W. Yan, Y. Song, Y. Zeng, M. Chen, L. Cheng, and Q. Liu, "Differentiate quality of experience scheduling for deep learning inferences with docker containers in the cloud," *IEEE Transactions on Cloud Computing*, 2022.

[8] Y. Mao, J. Wang, J. P. Cohen, and B. Sheng, "Pasa: Passive broadcast for smartphone ad-hoc networks," in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2014, pp. 1–8.

[9] J. Zhang, X. Zhao, Z. Chen, and Z. Lu, "A review of deep learning-based semantic segmentation for point cloud," *IEEE Access*, vol. 7, pp. 179 118–179 133, 2019.

[10] J. Wang, Y. Yao, Y. Mao, B. Sheng, and N. Mi, "Omo: Optimize mapreduce overlap with a good start (reduce) and a good finish (map)," in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, IEEE, 2015, pp. 1–8.

[11] E. Bisong, *Building machine learning and deep learning models on Google cloud platform: A comprehensive guide for beginners*. Apress, 2019.

[12] H. Harvey, Y. Mao, Y. Hou, and B. Sheng, "Edos: Edge assisted offloading system for mobile devices," in *The 26th International Conference on Computer Communications and Networks (ICCCN 2017)*, 2017.

[13] X. Liu, M. Yan, and J. Bohg, "Meteornet: Deep learning on dynamic 3d point cloud sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9246–9255.

[14] J. Wang, T. Wang, Z. Yang, Y. Mao, N. Mi, and B. Sheng, "Seina: A stealthy and effective internal attack in hadoop systems," in *International Conference on Computing, Networking and Communications*, 2017.

[15] J. Gao, H. Wang, and H. Shen, "Task failure prediction in cloud data centers using deep learning," *IEEE transactions on services computing*, 2020.

[16] W. Zheng, M. Tynes, H. Gorelick, Y. Mao, L. Cheng, and Y. Hou, "Flowcon: Elastic flow configuration for containerized deep learning applications," in *ACM the 48th International Conference on Parallel Processing (ICPP '19)*, 2019.

[17] B. A. Richards, T. P. Lillicrap, P. Beaudoin, Y. Bengio, R. Bogacz, A. Christensen, C. Clopath, R. P. Costa, A. de Berker, S. Ganguli, *et al.*, "A deep learning framework for neuroscience," *Nature neuroscience*, vol. 22, no. 11, pp. 1761–1770, 2019.

[18] W. Zheng, Y. Song, Z. Guo, Y. Cui, S. Gu, Y. Mao, and L. Cheng, "Target-based resource allocation for deep learning applications in a multi-tenancy system," in *2019 IEEE High Performance Extreme Computing Conference(HPEC '19)*, 2019.

[19] A. Acharya, Y. Hou, Y. Mao, M. Xian, and J. Yuan, "Workload-aware task placement in edge-assisted human re-identification," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON '19)*, IEEE, 2019, pp. 1–9.

[20] H. V. Pham, S. Qian, J. Wang, T. Lutellier, J. Rosenthal, L. Tan, Y. Yu, and N. Nagappan, "Problems and opportunities in training deep learning software systems: An analysis of variance," in *Proceedings of the 35th IEEE/ACM international conference on automated software engineering*, 2020, pp. 771–783.

[21] J. Wang, Y. Yao, Y. Mao, B. Sheng, and N. Mi, "Fresh: Fair and efficient slot configuration and scheduling for hadoop clusters," in *2014 IEEE 7th International Conference on Cloud Computing*, IEEE, 2014, pp. 761–768.

[22] A. Velloso and P. Van Hentenryck, "Combining deep learning and optimization for preventive security-constrained dc optimal power flow," *IEEE Transactions on Power Systems*, vol. 36, no. 4, pp. 3618–3628, 2021.

[23] Y. Mao, J. Oak, A. Pompili, D. Beer, T. Han, and P. Hu, "Draps: Dynamic and resource-aware placement scheme for docker containers in a heterogeneous cluster," in *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, IEEE, 2017, pp. 1–8.

[24] G. Muhammad and M. S. Hossain, "Emotion recognition for cognitive edge computing using deep learning," *IEEE Internet of Things Journal*, vol. 8, no. 23, pp. 16 894–16 901, 2021.

[25] Y. Mao, V. Green, J. Wang, H. Xiong, and Z. Guo, "Dress: Dynamic resource-reservation scheme for congested data-intensive computing platforms," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, IEEE, 2018, pp. 694–701.

[26] Y. Fu, S. Zhang, J. Terrero, Y. Mao, G. Liu, S. Li, and D. Tao, "Progress-based container scheduling for short-lived applications in a kubernetes cluster," in *2019 IEEE International Conference on Big Data (BigData'19)*, 2019.

[27] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu, Y. Mao, and J. Murphy, "A woa-based optimization approach for task scheduling in cloud computing systems," *IEEE Systems journal*, vol. 14, no. 3, pp. 3117–3128, 2020.

[28] Y. Mao, Y. Fu, W. Zheng, L. Cheng, Q. Liu, and D. Tao, "Speculative container scheduling for deep learning applications in a kubernetes cluster," *IEEE Systems Journal*, 2021.

[29] Y. Mao, Y. Fu, S. Gu, S. Vhaduri, L. Cheng, and Q. Liu, "Resource management schemes for cloud-native platforms with computing containers of docker and kubernetes," *arXiv preprint arXiv:2010.10350*, 2020.

[30] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "An autonomous computation offloading strategy in mobile edge computing: A deep learning-based hybrid approach," *Journal of Network and Computer Applications*, vol. 178, p. 102 974, 2021.

[31] L. Cheng, Y. Wang, Q. Liu, D. H. Epema, C. Liu, Y. Mao, and J. Murphy, "Network-aware locality scheduling for distributed data operators in data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1494–1510, 2021.

[32] Q. Liu, T. Xia, L. Cheng, M. Van Eijk, T. Ozcelebi, and Y. Mao, "Deep reinforcement learning for load-balancing aware network control in iot edge systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1491–1502, 2021.

[33] A. Yang, J. Wang, Y. Mao, Y. Yao, N. Mi, and B. Sheng, "Optimizing internal overlaps by self-adjusting resource allocation in multi-stage computing systems," *IEEE Access*, vol. 9, pp. 88 805–88 819, 2021.

[34] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 873–880.

[35] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv preprint arXiv:2007.05558*, 2020.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[37] D. So, Q. Le, and C. Liang, "The evolved transformer," in *International Conference on Machine Learning*, PMLR, 2019, pp. 5877–5886.

[38] S. A. Stein, B. Baheri, D. Chen, Y. Mao, Q. Guan, A. Li, S. Xu, and C. Ding, "Quclassi: A hybrid deep neural network architecture based on quantum state fidelity," *Proceedings of Machine Learning and Systems*, vol. 4, 2022.

[39] G. García-Pérez, M. A. Rossi, and S. Maniscalco, "Ibm q experience as a versatile experimental testbed for simulating open quantum systems," *npj Quantum Information*, vol. 6, no. 1, pp. 1–10, 2020.

[40] B. Baheri, D. Chen, B. Fang, S. A. Stein, V. Chaudhary, Y. Mao, S. Xu, A. Li, and Q. Guan, "Tqea: Temporal quantum error analysis," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, IEEE, 2021, pp. 65–67.

[41] A. P. Dash, S. K. Sahu, S. Kar, B. K. Behera, and P. K. Panigrahi, "Explicit demonstration of initial state construction in artificial neural networks using netket and ibm q experience platform," *Quantum Information Processing*, vol. 19, no. 1, pp. 1–15, 2020.

[42] S. A. Stein, R. L'Abbate, W. Mu, Y. Liu, B. Baheri, Y. Mao, G. Qiang, A. Li, and B. Fang, "A hybrid system for learning classical data in quantum states," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, IEEE, 2021, pp. 1–7.

[43] S. A. Stein, B. Baheri, D. Chen, Y. Mao, Q. Guan, A. Li, B. Fang, and S. Xu, "Qugan: A quantum state fidelity based generative adversarial network," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, IEEE, 2021, pp. 71–81.

[44] L. Zhao, S. Johri, J. Jakowski, T. Morris, R. Pooser, and D. Zhu, "Pushing the limit of quantum chemistry simulations with ion-trap hardware," *Bulletin of the American Physical Society*, 2022.

[45] L. Xue, L. Cheng, Y. Li, and Y. Mao, "Quantum machine learning for electricity theft detection: An initial investigation," in *2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, IEEE, 2021, pp. 204–208.

[46] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*, 2002.

[47] R. P. Feynman, "Quantum mechanical computers," *Foundations of physics*, vol. 16, no. 6, pp. 507–531, 1986.

[48] J. Patterson, "Coded character sets, history and development," *IEE Proceedings E-Computers and Digital Techniques*, vol. 128, no. 4, p. 173, 1981.

[49] N. S. Yanofsky and M. A. Mannucci, *Quantum computing for computer scientists*. Cambridge University Press, 2008.

[50] O. Morsch, *Quantum bits and quantum secrets: how quantum physics is revolutionizing codes and computers*. John Wiley & Sons, 2008.

[51] V. Kulkarni, M. Kulkarni, and A. Pant, "Quantum computing methods for supervised learning," *Quantum Machine Intelligence*, vol. 3, no. 2, pp. 1–14, 2021.

[52] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, Ieee, 1994, pp. 124–134.

[53] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[54] P. F. Windley, "Trees, forests and rearranging," *The Computer Journal*, vol. 3, no. 2, pp. 84–88, 1960.

[55] G. Brassard, "Searching a quantum phone book," *Science*, vol. 275, no. 5300, pp. 627–628, 1997.

[56] A. Mandviwalla, K. Ohshiro, and B. Ji, "Implementing grover's algorithm on the ibm quantum computers," in *2018 IEEE International Conference on Big Data (Big Data)*, IEEE, 2018, pp. 2531–2537.

[57] K. Kang, "Two improvements in grover's algorithm," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, IEEE, 2015, pp. 1179–1182.

[58] M. Sawerwain and M. Wróblewski, "Application of quantum k-nn and grover's algorithms for recommendation big-data system," in *International Conference on Information Systems Architecture and Technology*, Springer, 2018, pp. 235–244.

[59] I. Chakrabarty, S. Khan, and V. Singh, "Dynamic grover search: Applications in recommendation systems and optimization problems," *Quantum Information Processing*, vol. 16, no. 6, pp. 1–21, 2017.

[60] Y. Suzuki, S. Uno, R. Raymond, T. Tanaka, T. Onodera, and N. Yamamoto, "Amplitude estimation without phase estimation," *Quantum Information Processing*, vol. 19, no. 2, pp. 1–17, 2020.

[61] C. Wie, "Simpler quantum counting," *Quantum Inf. Comput.*, vol. 19, no. 11&12, pp. 967–983, 2019. DOI: 10.26421/QIC19.11-12-5. [Online]. Available: https://doi.org/10.26421/QIC19.11-12-5.

[62] S. Aaronson and P. Rall, "Quantum approximate counting, simplified," in *Symposium on Simplicity in Algorithms*, SIAM, 2020, pp. 24–32.

[63] D. Grinko, J. Gacon, C. Zoufal, and S. Woerner, "Iterative quantum amplitude estimation," *npj Quantum Information*, vol. 7, no. 1, pp. 1–6, 2021.

[64] A. Matuschak and M. A. Nielsen, "Quantum computing for the very curious," *URL: https://quantum. country/qcvc (cit. on p. 8)*, 2019.

[65] G. Lancaster, *Excel HSC Software Design & Development*. Pascal Press, 2001.

[66] W. I. Fletcher, *An engineering approach to digital design*. Prentice Hall PTR, 1997.

[67] F. Laloë, *Do we really understand quantum mechanics?* Cambridge University Press, 2019.

[68] L. Gurvits, "Classical deterministic complexity of edmonds' problem and quantum entanglement," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, 2003, pp. 10–19.

[69] A. Gilyén, S. Arunachalam, and N. Wiebe, "Optimizing quantum optimization algorithms via faster quantum gradient computation, 2017," *arXiv preprint arXiv:1711.00465*,

[70] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight bounds on quantum searching," *Fortschritte der Physik: Progress of Physics*, vol. 46, no. 4-5, pp. 493–505, 1998.

[71] *Most common words*, `https://en.wikipedia.org/wiki/Most_common_words_in_English`.