Grover's Algorithm for Closest-Value Search in Quantum Simulators

Bingying Liang
Southern Methodist University
Dallas, USA
bingyingl@smu.edu

***Abstract*—This study explores the use of Grover's algorithm on a quantum simulator to search for the element in an unordered array that is closest to a given target value. Grover's algorithm, as a core algorithm in quantum computing, theoretically provides a significant speedup for out-of-order search tasks. The goal is to verify the effectiveness of Grover's algorithm in finding values that are within a certain range of a specific number and to quantify its performance improvement over traditional linear search algorithms. By constructing a specific Oracle and the corresponding quantum circuit, Grover's algorithm is implemented on a quantum simulator, and experiments are carried out for a system with several qubits. The experimental results show that Grover algorithm can effectively increase the measurement probability of states close to the target value, and thus show better performance than traditional algorithms when searching unordered data sets. The research shows that Grover algorithm has a significant application prospect in the quantum computing environment, especially when dealing with the search problem of large-scale data sets. The results of this study provide new insights into the future application of quantum computing in the field of data processing and information retrieval, and open the way for further exploration of quantum search algorithms.**

***Keywords*—Grover's Algorithm, Quantum Computing, Quantum Search, Quantum Simulation,**

I. Introduction

In the era of Big Data, the ability to search through vast, unsorted databases efficiently stands as a critical challenge in the field of computer science. Even though classical algorithms have provided robust solutions for sorted data structures, searching unsorted arrays the time complexity of the best classical algorithms remains linear with respect to the size of the data set. Quantum computing brings the potential solution to this challenge. In 1982, Richard Feynman [16] presented the concept of Quantum Computation. Compared with classical computers the state of the unit is zero or one, the state of a quantum computer could be zero and one or anything in between. The unique characteristic of a quantum computer can create huge power of the parallel, which means it can pursue different paths at the same time during a single calculation unit [18]. Quantum computing presents a groundbreaking alternative with the potential to dramatically enhance search efficiency through quantum parallelism and interference.

Grover's algorithm, a quantum algorithm proposed by Lov Grover in 1996 [7]. Grover's algorithm offers a promising solution for searching unsorted databases, boasting a quadratic speedup over classical approaches by reducing the search complexity to $O(\sqrt{N})$ where N is the number of items. Despite Grover's algorithm being a significant theoretical advancement, its practical application is impeded by the current infancy of quantum hardware technology. This bottleneck has led researchers to leverage quantum simulators—classical systems designed to mimic quantum behavior—as a platform for exploring quantum algorithms. For example, the qiskit simulator platform [11] provides a controlled environment for testing, which is invaluable for understanding the practical aspects of quantum computational procedures and setting the stage for their eventual implementation on real quantum computers.

In the past few years, Grover's algorithm has shown the potential for some problems based on unstructured data. Aghaei [8] modified classical Dijkstra's algorithm based on Grover's algorithm which brings a more efficient algorithm. Y. Wang and M. Perkowski [19] show that on a graph coloring problem, the ternary oracle implemented in the multi-valued Grover Algorithm yields valid solutions because of the results from [20]. Jehn-Ruey Jiang [6] has pointed out that the well-known Grover algorithm can constructed with the explicit oracle to solve the Hamiltonian cycle problem for the complete graph because the quantum circuit has a quadratic speedup over the classical unstructured search algorithm for solving the same problem.

Grover's algorithm is also known as a quantum search algorithm [5], which enables this search method to be sped up substantially. It's quite important for the unsorted array problems. During the computer network, especially the link layer, such as a sliding window protocol using selective repeat, the packet transmission information is stored in the arrays [3]. When one of the packets is lost, the receiver has to use linear time to check which packet is lost. Even some classical array search algorithm like binary search [2] needs to sort the array first. So the unsorted array search is an important basement of other algorithms. And depending on the quantum parallelism, it can improve the efficiency of the unsorted array search.

The structure of this paper is as follows: Section 2 provides a detailed background on classical search methods and Grover's Algorithm. Section 3 outlines the methodology for applying Grover's Algorithm in a quantum simulator to enhance unsorted array search efficiency, particularly in finding the closest value. Section 4 discusses the results and analysis of this approach. Finally, Section 5 concludes the paper with key findings and implications.

II. Background

A. Limitations of classical search

Classical search, also known as linear search or sequential search, is an elementary search algorithm. The input is unsorted. The algorithm compares the values one by one until a match is found. The time complexity of the linear search is $O(n)$ where n is the size of the array [2]. The best case of linear search is when the target value is at the first position of the array. The worst case of linear search is when the target value is at the end position of the array or the value is not in the array, which means the algorithm needs to traverse the entire array to search the value. The linear search algorithm is the following:

Algorithm 1 Classical Unsorted Search (Linear Search)

```
Input: Array $A[x_0, x_1, ..., x_{n-1}]$, Value to find $v$
Output: Index of $w$ in Array $A$ or null (if $w$ is not in $A$)
    Initialisation :
 1: $index \leftarrow$ null
    Search Process :
 2: for $i = 0$ to $n - 1$ do
 3:     if $(A[i] = w)$ then
 4:         $index \leftarrow i$
 5:         break Value found, exit loop
 6:     end if
 7: end for
 8: return $index$
```

B. Classical closest value search

In addition to finding an exact match, another common need when dealing with unordered arrays is to find the element that is closest to a given target value [4]. In traditional computations, this typically requires traversing the entire array to determine the closest element to the target value. Compared to finding an exact match, finding the closest value involves an additional computational step, as each element needs to be compared to the target value.

For the nearest value search of an unordered array, the time complexity of the algorithm is still $O(n)$, where n is the size of the array. The basic process of the algorithm is as follows: This algorithm first initializes a variable to record the minimum difference ($minDiff$) and an index variable ($index$). It then iterates over each element in the array, computes the difference between each element and the target value t, and updates the minimum difference and its corresponding index. After the traversal is complete, the algorithm returns the index of the element closest to the target value. In practice, especially on large datasets, this linear approach may not be efficient. Therefore, exploring more efficient algorithms, such as methods using quantum computing, may provide faster solutions.

Algorithm 2 Classical Closest Value Search in Unsorted Array

```
Input: Array $A[x_0, x_1, ..., x_{n-1}]$, Target value $t$
Output: Index of the closest value to $t$ in Array $A$
 1: $minDiff \leftarrow \infty$
 2: $index \leftarrow -1$
    Search Process :
 3: for $i = 0$ to $n - 1$ do
 4:     $diff \leftarrow |A[i] - t|$
 5:     if $diff < minDiff$ then
 6:         $minDiff \leftarrow diff$
 7:         $index \leftarrow i$
 8:     end if
 9: end for
10: return $index$
```

C. Grover's Algorithm

Grover's algorithm [7] is a quantum algorithm proposed by Grover in 1996 to solve the unstructured search problem with a high probability. Suppose in $N = 2^n$ do the search. The algorithm is summarized in the following:

1) Initialize a quantum system of $n + 1$ qubits with state $|0\rangle$.
2) Apply H gate to the first n qubits, and XH to the last qubit.
3) Repeat the below steps in Grover iteration $G \approx \left\lceil \frac{\pi\sqrt{2^n}}{4} \right\rceil$ times:
   a) Apply an Oracle's operation
   $$f(x) = \begin{cases} 0, \text{ if } x \neq w \\ 1, \text{ if } x = w \end{cases} \quad (1)$$
   Oracle
   $$U_f = (-1)^{f(x)} |i\rangle \quad (2)$$
   which is also called the black box. It flips the amplitude of the desired state.
   b) Apply the diffusion operator
   $$U_s = H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} \quad (3)$$
   $$= 2|s\rangle\langle s| - I \quad (4)$$
   where $|s\rangle$ is the equally weighted superposition. The operator performs an inversion by the mean, which apples on all amplitudes.
4) Measure the first n qubits.

Fig. 1 shows the circuit diagram [15] for Grover's algorithm and the pseudocode of algorithm [5] [15] is in Algorithm 2.

For unsorted array search, Grover's Algorithm is particularly relevant as it does not necessitate any preliminary sorting or structuring of data. The algorithm effectively squares the search speed, which is profound for large datasets where classical algorithms falter. Grover's Algorithm also benefits from the intrinsic properties of quantum mechanics such as superposition and entanglement, which enables the quantum computer to evaluate multiple states simultaneously, further contributing to its search efficiency.

Algorithm 3 Grover's Algorithm

```
Input: A black-box oracle $O$ that marks the winner state $|w\rangle$,
    number of elements $N$
Output: The winner state $|w\rangle$
    Initialisation :
1:  Prepare a uniform superposition of all states, $\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle$
    Oracle and Amplification :
2:  for $k = 1$ to $approx \left\lceil \frac{\pi\sqrt{2^n}}{4} \right\rceil$ times do
3:      Apply the oracle $G$ to mark the winner state $|w\rangle$
4:      Apply the Grover diffusion operator $U_s$ for amplitude
        amplification
5:  end for
    Measurement :
6:  Measure the quantum state to obtain the winner state $|w\rangle$
7:  return The winner state $|w\rangle$ or null if not found
```

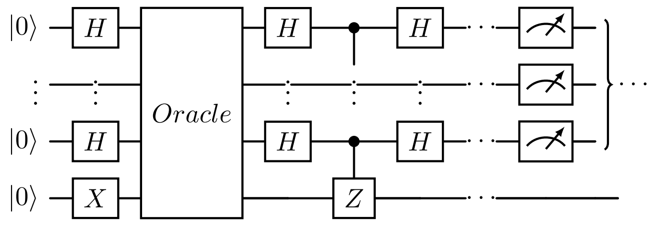

Fig. 1. Grover's Algorithm Circuit

D. *The evolution of quantum simulators*

Quantum simulators have played a pivotal role in the development and testing of quantum algorithms. The progression from rudimentary simulators capable of handling only a few qubits to today's sophisticated systems marks a significant milestone in quantum computing. In the past few years, scores of frameworks, tools, and platforms are emerged, and improvement of currently available facilities would exploit the research activities in the quantum research community. Paramita Basak Upama [14] did a survey on these frameworks, which mentions IBM's 5 qubits gate-level quantum processor on the web that allows the users to apply to get access to it. Quantum simulators like IBM's Qiskit simulator platforms support the execution of quantum programs within a controlled environment.

Qiskit [10] is a framework developed by IBM, which allows users to create quantum programs and run them either on a simulation environment or IBM's real quantum devices. The normal quantum program in Qiskit usually is built in two steps: First, users should create a quantum circuit which requires qubits and classical bits. The purpose of the classical bits is to store the results of measurements that will be taken later. Secondly, various gates and measurements need to be added to the circuit [11]. This paper will use Qiskit to implement algorithms.

III. Approach

A. *Theoretical Analysis*

1) Qubit Mappiing: Maurice Clerc 's [9] and N. Khammassi [12], can use mathematical ways to find some relation between classical bits and quantum qubits, which is also called qubit mapping. For example, the array is $\{1, 0, 2, 3\}$. Then it also be written in binary sequence, like $\{01, 00, 10, 11\}$. In binary sequence, it will be quite a simple mapping from bits to qubits. In this way, the array actually is very convenient for mapping to the qubits and also convenient for Grover's algorithm's application.

The unsorted array search problem, in the realm of classical computing, entails scanning each element of the array until the desired value is located, typically requiring linear time complexity. Due to the mapping, this problem is transformed by Grover's Algorithm, which uses the principles of superposition and interference to search all items simultaneously, theoretically yielding a quadratic speedup with a square root time complexity. The algorithm's unique approach to processing information encapsulates the potential of quantum computing to outpace classical methods in specific computational tasks.

2) Theoretical Quantum Speedup: In the example above, in a classical machine, it would have to run the code 4 times when searching for "3" in the array. However, in a quantum machine, it can search the entire array of elements simultaneously. Therefore, it only needs to be run once. As the number of inputs increases, the speedup effect of the quantum machine becomes more pronounced. Because the Grover algorithm is based on the amplitude amplification phenomenon of quantum mechanics, its search speed is significantly improved compared with the classical algorithm. While classical search requires $O(N)$operations to find an element in an unordered array of Nitems, Grover's algorithm can locate an item in about $O(\sqrt{N})$operations. This quadratic speedup is particularly beneficial when the size of the dataset increases, in which case classical algorithms become impractical. This further extended to find the closest to the value variation, not only to mark a single state of "the winner" $|w\rangle$, but also to tag all with $|w\rangle$ status within δapart. Here is the pseudocode for the corresponding algorithm 4. The goal is to find the state of the "winner" $|w\rangle$ within a given distance δ difference state of all. This means that the Oracle labels not only a single winner state but also all states that are close to the winner state. This requires the Oracle to be able to identify and label these close states, which can be more complicated than the Oracle in the original Grover algorithm

B. *Resources*

This approach leverages open-source quantum programming frameworks for the implementations. The primary platform will be IBM's Qiskit [10], given its compatibility with IBM's quantum machines. Access to high-performance classical computers will be essential for comparison benchmarks [13]. Compared with C-style languages, Python is easier for beginners, Qiskit supports Python, which means Qiskit may be more appropriate for beginners. [17]

Algorithm 4 Modified Grover's Algorithm for Closest Value Search

```
Input: An oracle $O$ that marks states close to the winner state
    $|w\rangle$, number of elements $N$, and distance $\delta$
Output: State(s) closest to the winner state $|w\rangle$ within distance $\delta$
    Initialisation :
1:  Prepare a uniform superposition of all states, $\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle$
    Oracle and Amplification :
2:  for $k = 1$ to $approx \left\lceil \frac{\pi\sqrt{2^n}}{4} \right\rceil$ times do
3:      Apply the modified oracle $O$ to mark states $|s\rangle$ such
        that $|s - w| \leq \delta$
4:      Apply the Grover diffusion operator $U_s$ for amplitude
        amplification
5:  end for
    Measurement :
6:  Measure the quantum state to obtain state(s) close to $|w\rangle$
7:  return State(s) close to $|w\rangle$ or null if not found
```

The resources required for this approach include access to quantum simulators (either local or cloud-based), a development environment with Qiskit installed, and computational tools for data analysis and visualization. Additionally, sufficient computational power is necessary to handle the simulation of larger quantum circuits as the array size grows.

C. Experimental Design

This experiment aims to simulate Grover's algorithm on the quantum simulator [1] by Qiskit to verify its effectiveness in finding elements in an unordered array that differ from a target value by a specific distance, δ. A quantum circuit with 4 qubits will be built in order to be able to represent integers between 0 and 15 and also to change the number of qubits and the target value as well as δ. The results are analyzed to observe their advantages in terms of speed or resource utilization.

1) Objective: The goal of the experiment is to verify the efficiency of Grover's algorithm on the quantum simulator for finding values in an unordered array that are at a specific distance from a given target value. In particular, wanted to show the performance of the algorithm in determining the proximity of the target value and the potential advantages over traditional search algorithms.

2) Experimental Setup: The experiments will use Qiskit, an open-source quantum computing framework, to build and run Grover's algorithm. Built a quantum circuit that implements the Oracle of Grover's algorithm along with the diffusion operation and applies it to a qubit system of a specific size to simulate the search process of finding an element in an unordered array that differs by one from a target value. The resource estimation relates to the number of qubits required for the experiment, the running time of the simulator, and the computational resources required for data analysis. This experiment requires 4 qubits, which is enough to cover 16 possible search states. Using the "qasm_simulator" provided by Qiskit, which can efficiently simulate this quantum circuit on a typical desktop or laptop computer.

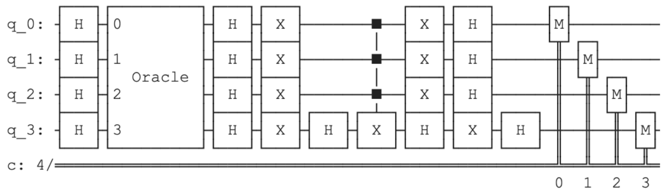

Fig. 2. Circuit for 4 qubits

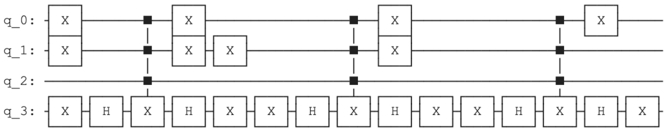

Fig. 3. Oracle for 4 qubits, $\delta = 1$

3) Procedure: The experimental procedure is as follows:

- Initialize the quantum circuit: Create a quantum circuit with four qubits and four classical bits, and put all the qubits in superposition. Figure 2 shows the circuit for 4 qubits.
- Build Oracle: Create an Oracle for the target value and its close values. Figure 3 4 shows oracle.
- Grover Iteration: Grover iteration is implemented in the circuit, including Oracle and diffusion operations, the pseudocode is shown in figure 5.
- Measurement and repetition: The quantum circuit is measured the results are recorded, and the experiment is repeated enough times to obtain statistically significant results.
- Data analysis: The collected results are used to analyze the performance of the algorithm and compare it with classical algorithms.

Also uses 5 qubits to do the same experiment. The Fig 6 shows the circuit for 5 qubits. In this way, it is expected to demonstrate the effectiveness and potential performance benefits of Grover's algorithm on the problem of finding the closest value in an unordered array.

IV. Results

The purpose of this experiment is to explore the performance of Grover's algorithm when searching for the element closest to a given target value in an unordered array on a quantum simulator. Here, it sets the target value to 5 (binary representation of 0101) and considers the case of differences of 1 and 2, corresponding to the search interval, respectively

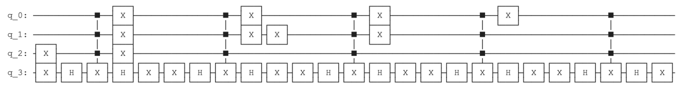

Fig. 4. Oracle for 4 qubits, $\delta = 2$

```
def create_oracle(target, n, delta):
    oracle_qc = QuantumCircuit(n)
    for d in range(-delta, delta + 1):
        value = (target + d) % (2**n)
        value_bin = format(value, '0' + str(n) +
        ↪  'b')
        for qubit, bit in
        ↪  enumerate(reversed(value_bin)):
            if bit == '0':
                oracle_qc.x(qubit)
        oracle_qc.h(n-1)
        oracle_qc.mct(list(range(n-1)), n-1)
        oracle_qc.h(n-1)
        for qubit, bit in
        ↪  enumerate(reversed(value_bin)):
            if bit == '0':
                oracle_qc.x(qubit)
    oracle_gate = oracle_qc.to_gate()
    oracle_gate.name = "Oracle"
    oracle_qc.draw()
    return oracle_gate

def grover_iteration(circuit, oracle, n):
    circuit.append(oracle, range(n))
    circuit.h(range(n))
    circuit.x(range(n))
    circuit.h(n-1)
    circuit.mct(list(range(n-1)), n-1)
    circuit.h(n-1)
    circuit.x(range(n))
    circuit.h(range(n))
    return circuit
```

Fig. 5. Pseudocode of oracle and iteration

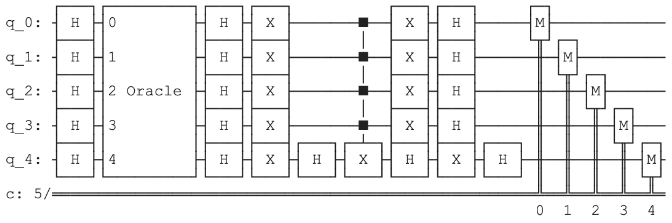

Fig. 6. Circuit for 5 qubits

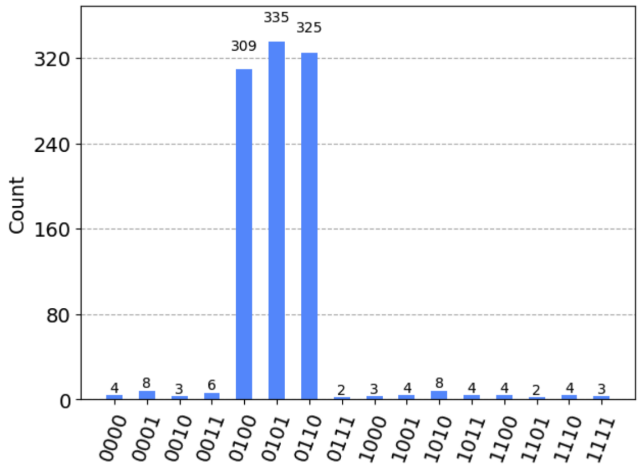

Fig. 7. target $= 5(0101)$, $\delta = 1$

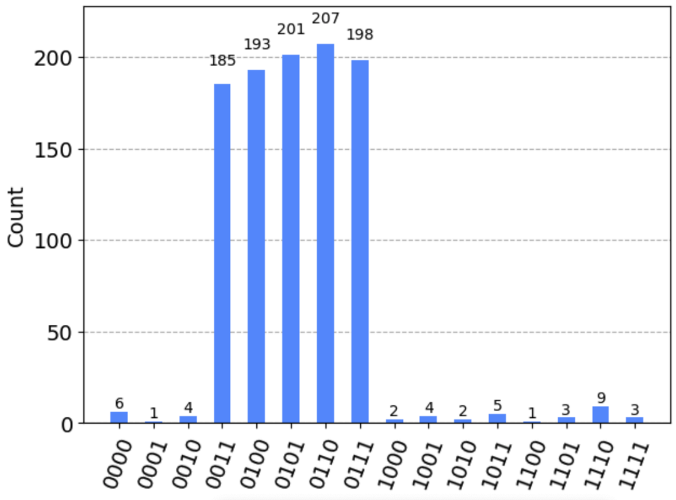

Fig. 8. target $= 5(0101)$, $\delta = 2$

$[4, 6]$ and $[3, 8]$. And then increase one qubit to 5 qubits, which sets the target value to 5 (binary representation of 00101, and also sets the $\delta = 1$ and $\delta = 2$.

A. Performance

In the case of a difference of 1, it was observed that Grover's algorithm increases the probability that the states '0100' (4 in decimal) and '0110' (6 in decimal) are measured. This is in line with our expectations since these states are the values adjacent to the target value of 5. The experimental results in figure 7 show that these two states are measured with high frequency and account for a significant part of all measurements.

When the difference is increased to 2, that is, the search interval is extended to $[3, 8]$, it was found that in addition to states 0100 and 0110, the measurement frequency of states 0011 (3 in decimal) and 1000 (8 in decimal) also increases accordingly see in figure 8. This indicates that the Oracle construction is successful in labeling states in a wider range around the target value.

For the 5 qubits, the results shown in figure9 and figure10

B. Comparison with Classical Search

Grover's algorithm shows clear advantages over conventional linear search. When executed on the quantum simulator, Grover's algorithm requires significantly fewer iterations than a classical search over the entire dataset. Specifically, the number of iterations of Grover's algorithm is approx $\sqrt{N}$, where N is the size of the search space, while the number of iterations of the linear search N

C. Resource Utilization

In terms of resource utilization, the quantum resource requirement of Grover's algorithm is linear in the number of qubits. For a 4-qubit system and 5-qubit system, it is possible to simulate running Grover's algorithm in a standard desktop

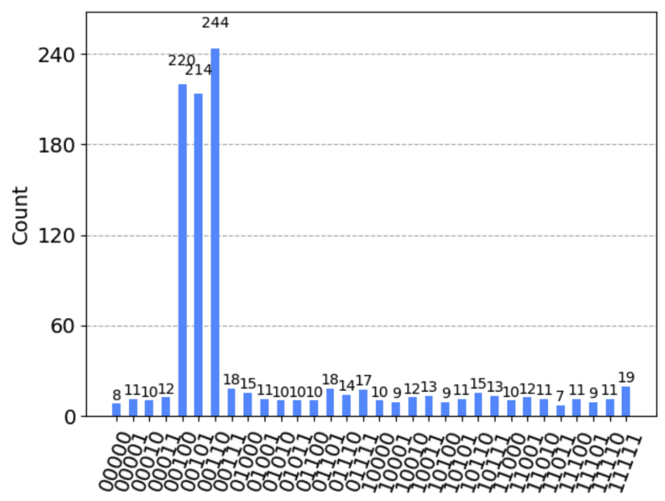

Fig. 9. target $= 5(00101)$, $\delta = 1$

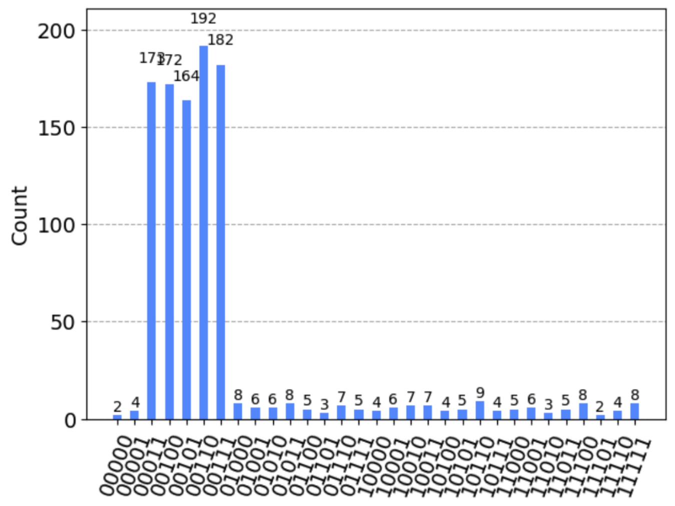

Fig. 10. target $= 5(00101)$, $\delta = 2$

computing environment without significant computational resources. In contrast, although the classical search algorithm has low resource consumption, its performance is far worse than Grover's algorithm on large data sets.

D. Experimental Observations

In the experimental observations, it is noted that the performance of Grover's algorithm is affected by several factors, including the accurate construction of the Oracle, the correct implementation of the diffusion operation, and the accurate measurement of the qubits. No unexpected error patterns were observed in the experiments, which further demonstrates the reliability of the quantum simulator as an experimental platform.

V. Summary

In this study, Grover's algorithm was successfully implemented on a quantum simulator to search for elements in an unordered array that differ by a specific range from a given target value. Experimental results show that Grover's algorithm can effectively increase the probability that states that are close to the target value are measured, thus providing significant performance gains when searching unordered datasets.

Through this study, the application potential of Grover's algorithm in the quantum computing environment is demonstrated, especially when dealing with the search problem of large-scale data sets. Grover's algorithm shows significant speedup compared to the traditional linear search algorithm, which provides strong evidence for the future application of quantum computing in the field of data processing and information retrieval.

In addition, the innovative design of Oracle in the experiment shows the flexibility and adaptability of quantum algorithms in solving practical problems. By adjusting the Oracle to label states that have a specific difference from the target value, can explore the application of Grover's algorithm in a wider range of search scenarios.

Despite the positive results of this study, some challenges in the implementation of quantum algorithms are revealed, especially in Oracle construction and circuit optimization. Future work can focus on the following areas:

1) Further optimization of Oracle: Explore more efficient ways to build an Oracle to be able to handle more complex search criteria.
2) Algorithm Scalability Study: Tested Grover's algorithm in larger-scale quantum systems to evaluate its performance on real quantum hardware.
3) Comparison with other quantum algorithms: The performance comparison of Grover's algorithm with other quantum search algorithms is investigated to determine the respective advantages and application scenarios.

Overall, the findings of this study provide valuable insights into the application of quantum computing to search and data processing and lay the foundation for future research in this exciting area.

References

[1] 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), Seattle, WA, USA: IEEE, Dec. 2018, pp. 2531–2537. doi: 10.1109/BigData.2018.8622457.

[2] A. E. Jacob, N. Ashodariya, and A. Dhongade, "Hybrid search algorithm: Combined linear and binary search algorithm," in 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai: IEEE, Aug. 2017, pp. 1543–1547. doi: 10.1109/ICECDS.2017.8389704.

[3] A. S. Tanenbaum, N. Feamster, and D. Wetherall, Computer networks, Sixth edition, Global edition. Harlow, United Kingdom: Pearson, 2021.

[4] 'Closest value to K from an unsorted array', GeeksforGeeks. Accessed: Dec. 04, 2023. [Online]. Available: https://www.geeksforgeeks.org/closest-value-to-k-from-an-unsorted-array/

[5] I. L. C. Michael A. Nielsen, Quantum Computation And Quantum Information, 10th Anniversary Edition. Cambridge University Press, 2010.

[6] J.-R. Jiang, "Quantum Circuit Based on Grover Algorithm to Solve Hamiltonian Cycle Problem," in 2022 IEEE 4th Eurasia Conference on IOT, Communication and Engineering (ECICE), Yunlin, Taiwan: IEEE, Oct. 2022, pp. 364–367. doi: 10.1109/ECICE55674.2022.10042919.

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

[8] M. R. S. Aghaei, Z. A. Zukarnain, A. Mamat, and H. Zainuddin, "A Hybrid Algorithm for the Shortest-Path Problem in the Graph," in 2008 International Conference on Advanced Computer Theory and Engineering, Phuket, Thailand: IEEE, Dec. 2008, pp. 251–255. doi: 10.1109/ICACTE.2008.137.

[9] M. Clerc, "Graph colouring: a polynomial complexity quantum algorithm," 2023, doi: 10.13140/RG.2.2.36154.77760.

[10] M. S. A. et al., "Qiskit: An open-source framework for quantum computing," 2021.

[11] M. Kashif and S. Al-Kuwari, "Qiskit As a Simulation Platform for Measurement-based Quantum Computation," in 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C), Honolulu, HI, USA: IEEE, Mar. 2022, pp. 152–159. doi: 10.1109/ICSA-C54293.2022.00037.

[12] N. Khammassi, I. Ashraf, X. Fu, C. G. Almudever, and K. Bertels, "QX: A high-performance quantum computer simulation platform," in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, Lausanne, Switzerland: IEEE, Mar. 2017, pp. 464–469. doi: 10.23919/DATE.2017.7927034.

[13] "New IBM, UC Berkeley paper shows path toward useful quantum," IBM Research Blog. Accessed: Nov. 06, 2023. [Online]. Available: https://research.ibm.com/blog/utility-toward-useful-quantum

[14] P. B. Upama et al., "Evolution of Quantum Computing: A Systematic Survey on the Use of Quantum Computing Tools," in 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, CA, USA: IEEE, Jun. 2022, pp. 520–529. doi: 10.1109/COMPSAC54236.2022.00096.

[15] R. H. Preston, "Applying Grover's Algorithm to Hash Functions: A Software Perspective," IEEE Trans. Quantum Eng., vol. 3, pp. 1–10, 2022, doi: 10.1109/TQE.2022.3233526.

[16] R. P. Feynman, "Simulating physics with computers," Int. J. Theor. Phys., vol. 21, no. 6–7, pp. 467–488, 1982.

[17] R. LaRose, "Overview and Comparison of Gate Level Quantum Software Platforms," Quantum, vol. 3, p. 130, Mar. 2019, doi: 10.22331/q-2019-03-25-130.

[18] S. B. Ramezani, A. Sommers, H. K. Manchukonda, S. Rahimi, and A. Amirlatifi, "Machine Learning Algorithms in Quantum Computing: A Survey," in 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, United Kingdom: IEEE, Jul. 2020, pp. 1–8. doi: 10.1109/IJCNN48605.2020.9207714.

[19] Y. Wang and M. Perkowski, "Improved Complexity of Quantum Oracles for Ternary Grover Algorithm for Graph Coloring," in 2011 41st IEEE International Symposium on Multiple-Valued Logic, Tuusula, Finland: IEEE, May 2011, pp. 294–301. doi: 10.1109/ISMVL.2011.42.

[20] Y. Fan, "Applications of Multi-Valued Quantum Algorithms," in 37th International Symposium on Multiple-Valued Logic (ISMVL'07), May 2007, pp. 12–12. doi: 10.1109/ISMVL.2007.3.