INDIAN INSTITUTE OF SCIENCE

IISC QUANTUM TECHNOLOGY INITIATIVE

QT312 - ADVANCED QUANTUM TECHNOLOGY LABORATORY

Project Report

Implementation of Grover's Search Algorithm

Chaitali Shah
(SR No: 01-02-04-10-51-21-1-19786)

Date: February 15, 2022

Contents

1 A Brief Introduction to Grover's Algorithm

Grover proposed a quantum algorithm to solve the **unstructured database search problem**. Its computational complexity is of order $O(\sqrt{N})$, which provides a quadratic speed-up over its classical counterparts, that have a complexity of order $O(N)$, where N is the size of the database/list.

The N items of the database are mapped to an orthonormal basis $|i\rangle, i = 1, 2, ...N$ in an N-dimensional Hilbert space. The search problem is framed as finding a particular target state from an initial state, using an oracle. An **oracle** is a black-box that can answer a binary question. In the search algorithm, a single question can be asked at every iteration until the target item is obtained. The quantum advantage lies in the ability to use the oracle on a *superposition* of states rather than a single state of the basis.

Grover's Algorithm consists of the following steps:

1. **Initialisation:** Since there is no information regarding the structure of the database, the probability that any vector is the target vector is $\frac{1}{N}$. The initial state, denoted by $|s\rangle$ is an unbiased uniform superposition state, with components of equal amplitude $\frac{1}{\sqrt{N}}$ along each of the basis vectors, as shown in Fig. 1. The required state, henceforth called the target state $|t\rangle$, is also a basis vector.

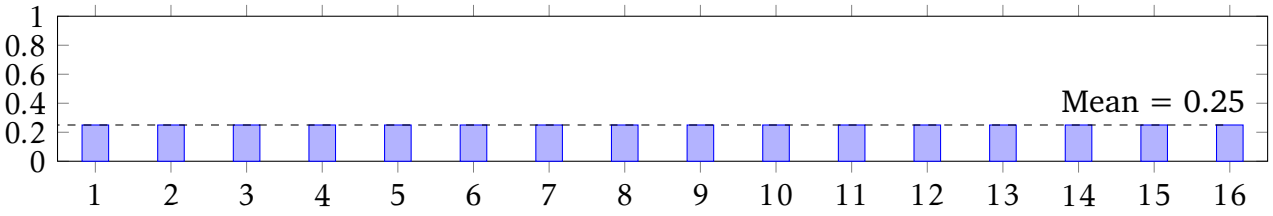


Figure 1: Wave picture corresponding to the initialisation - Amplitude of each mode is $\frac{1}{\sqrt{N}} = \frac{1}{4}$

2. **The Oracle:** The oracle is a unitary operation in the Hilbert space that multiplies -1 to the vector if is the target state vector, and multiplies +1 for any other basis vector as shown in Fig. 2. Such an operation is implemented by the unitary transformation $U_t = I - 2|t\rangle\langle t|$. This operation reflects the initial state about the direction perpendicular to $|t\rangle$, so that the initial vector is rotated by 2θ, where θ is the angle between vector $|s\rangle$ and the vector perpendicular to $|t\rangle$ in the $|s\rangle$-$|t\rangle$ plane (refer Fig. 14).

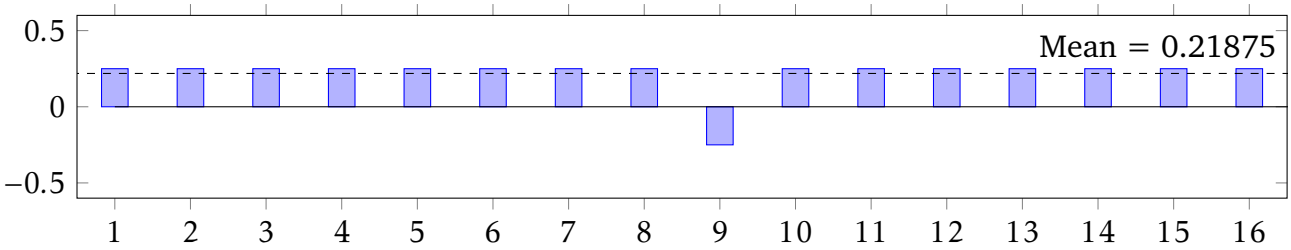


Figure 2: Wave picture corresponding to the oracle query- the amplitude of the target, the 10^{th} mode, is flipped in sign

3. **Amplitude Amplification or Diffusion:**

   After the oracle query, information is gained about the target state in terms of a change in sign. Now, a reflection about the vector $|s\rangle$ is performed using the unitary transformation $-U_s = 2|s\rangle\langle s| - I$. This step is called **amplitude amplification** because the amplitude of the target state is increased by performing a reflection about the mean (Fig. 3). $-U_s$ is also called a **diffusion** matrix since it can be interpreted as a probability matrix and the sum of all elements along a row or a column add is 1. This operation results in the state vector being rotated such that it is at an angle θ from the initial state $|s\rangle$ towards the state $|t\rangle$ in the $|s\rangle$-$|t\rangle$ plane (refer Fig. 14).

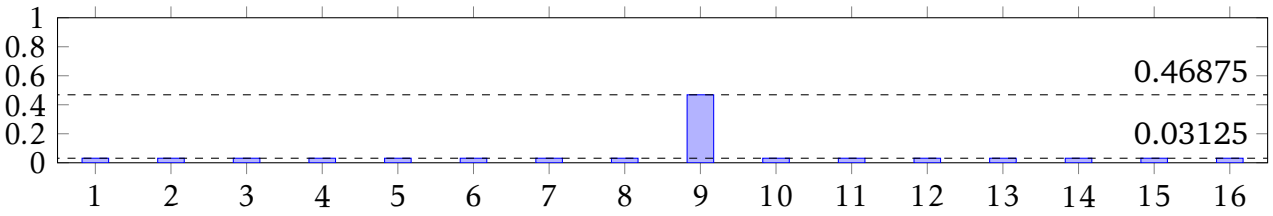


Figure 3: Wave picture corresponding to the amplitude amplification - Amplitudes are reflected about the mean

4. **Measurement:** Each iteration of the Grover's algorithm involves the application of the operation $(-U_s U_t)$ to the state vector until it is sufficiently close to $|t\rangle$. For a large database of size N, the number of iterations required is $Q \approx \frac{\pi}{4}\sqrt{N}$. After the required iterations, the amplitude of the target state is high and hence the probability of obtaining the target state on subsequent measurement is also high.

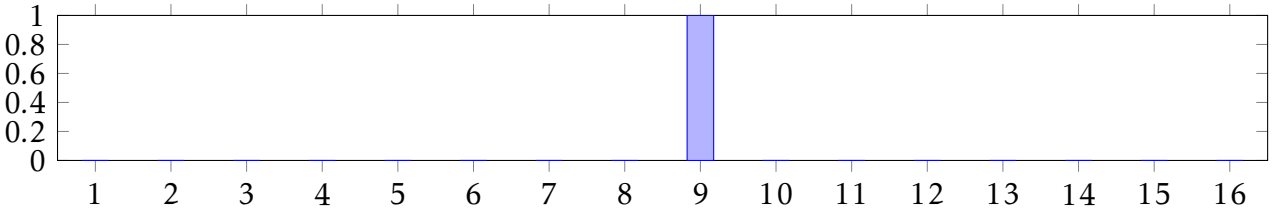


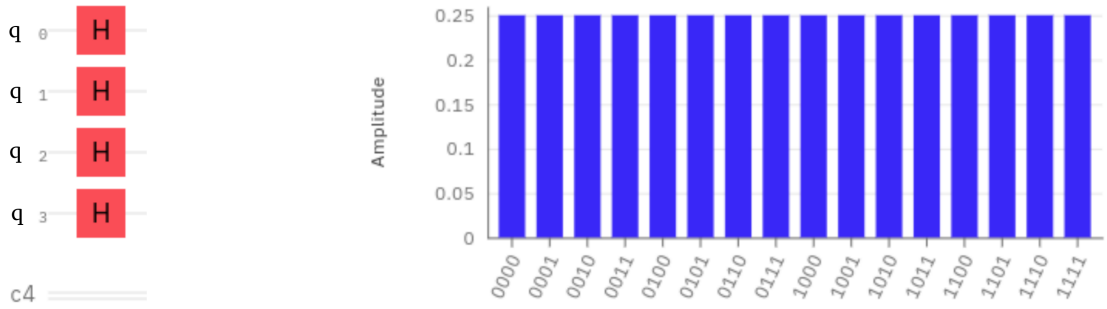
Figure 4: Wave picture corresponding to measurement - The mode with highest probability is found to be the target mode

2 Implementation of the Algorithm

Grover's Algorithm for 4 qubits is implemented in Qiskit as shown in the following sections.

2.1 Initialisation

A uniform superposition state can be obtained by using the Hadamard operator, H on all the qubits that are initialised to $|0\rangle$: $|s\rangle = H^{(4)}|0\rangle$

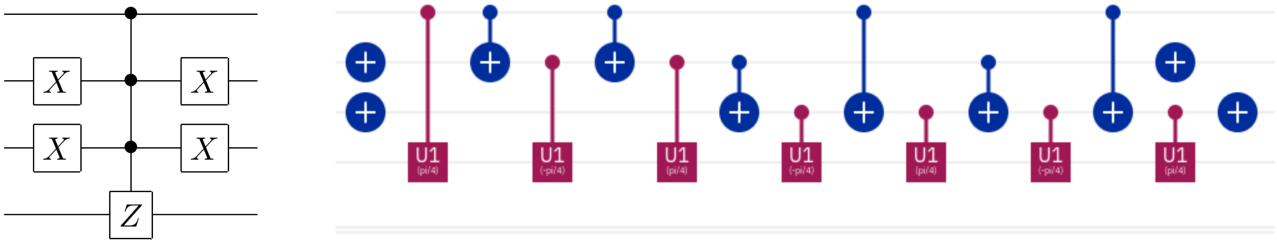


(a) (b)

Figure 5: (a) The quantum circuit implementation using H gates in Qiskit (b) Amplitude of the state vector of the initial state

2.2 Oracle query

The oracle is given by the unitary operation $U_t = I - |t\rangle\langle t|$.
In matrix notation it is a diagonal matrix with all diagonal elements 1, except the diagonal element corresponding to the target state which is -1. This operation can be performed by a controlled-Z gate, along with X gates depending on the target state. The target state in this case is 1001 an the oracle is implemented as shown in Fig. 6.



(a) (b)

Figure 6: (a) The oracle for 1001 using H and controlled-Z gates (b) The quantum circuit implementation using X, controlled-X and CU1 gates in Qiskit

The flipping of sign is shown by a phase of π in the plot of amplitude in Fig. 7. The red colour indicates that the phase of the state is π.

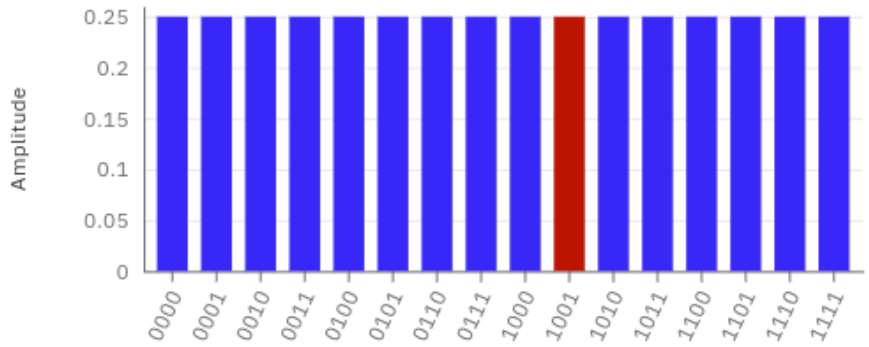


Figure 7: Amplitude of the state vector of the state after the oracle query

2.3 Amplitude Amplification

Amplitude amplification is performed by the matrix $-U_s = 2|s\rangle\langle s| - I$. $|s\rangle$ is a column vector of size $\lceil \log(N) \rceil$, with all entries equal to $\frac{1}{\sqrt{N}}$. All the entries of the matrix $|s\rangle\langle s|$ are equal to $\frac{1}{N}$. Therefore, the matrix $-U_s$ has diagonal elements $\frac{2}{N} - 1$ and off-diagonal elements $\frac{2}{N}$.

$$|s\rangle = H^{(4)}|0\rangle \implies 2|s\rangle\langle s| - I = H^{(4)}(2|0\rangle\langle 0| - I)H^{(4)}$$

$$-U_s = H^{(4)}(2|0\rangle\langle 0| - I)H^{(4)} = -H^{(4)}U_0H^{(4)}$$

The $-U_0$ operation can be implemented by X gates and controlled Z gate. The diffuser circuit is constructed as shown in Fig 8.

Figure 8: (a) The amplitude amplifier circuit (b) The quantum circuit implementation using H, controlled X and CU1 gates in Qiskit

The amplitude amplification of the target state can be seen in the Fig. 9.

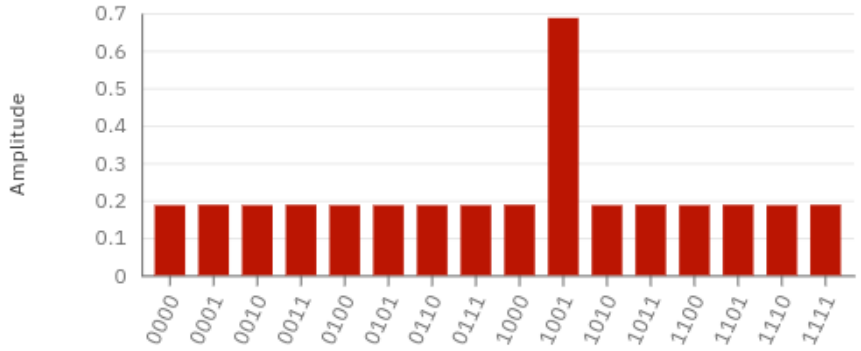
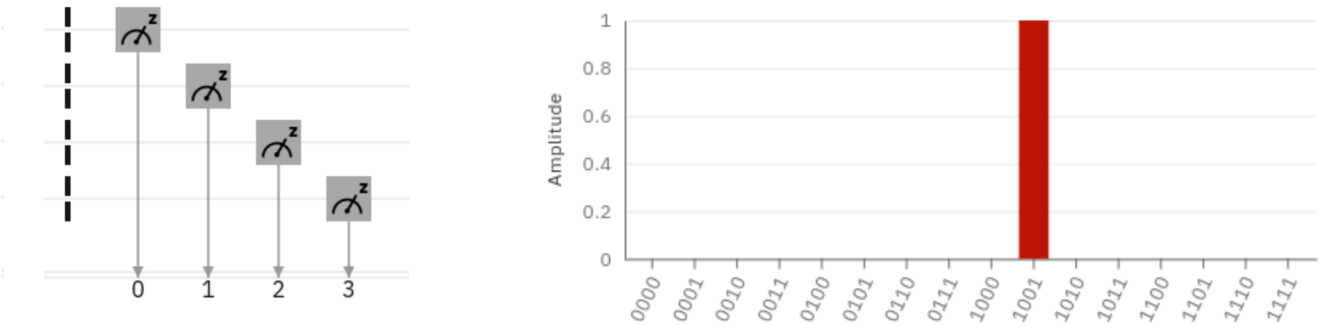


Figure 9: Amplitude of the state vector of the state after amplitude amplification

2.4 Measurement

For N=16, the number of iterations required is Q = 3. After 3 iterations, the measurement of the state is performed as shown below. A barrier is added before measurement to avoid any reordering of gates due to back-end optimization.



(a) (b)

Figure 10: (a) The measurement circuit (b) The target state is obtained as a result of the measurement

3 Results

3.1 Probability of states

The plots of probability of states after each iteration of the Grover's algorithm are shown below.

3.1.1 First Iteration

Probability of obtaining the correct target state 1001 in (a) Simulator = 0.490 and (b) Real Device = 0.102.

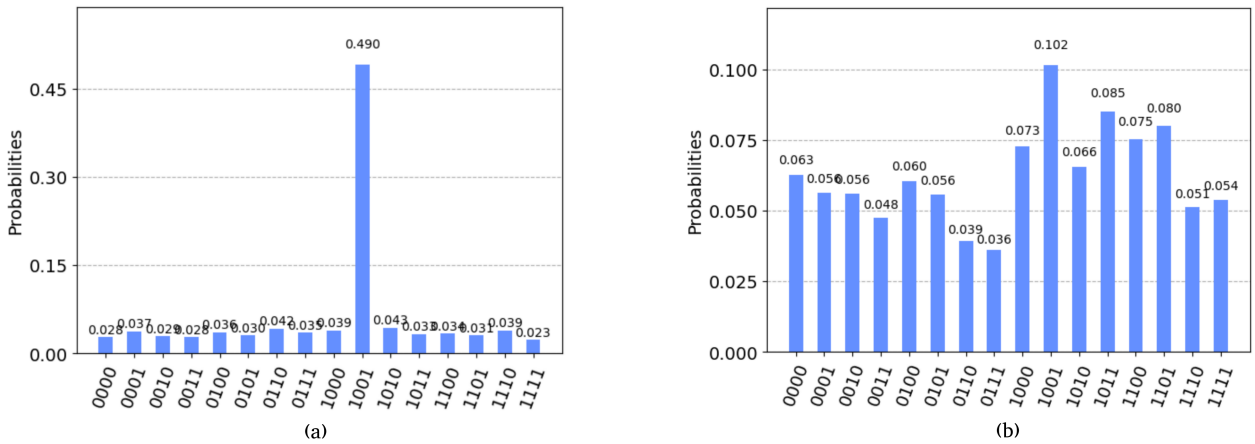


Figure 11: Iteration 1 : (a) Simulator : qasm_simulator (b) Real Quantum Device : ibmq_lima

3.1.2 Second Iteration

Probability of obtaining the correct target state 1001 in (a) Simulator = 0.924 and (b) Real Device = 0.078.

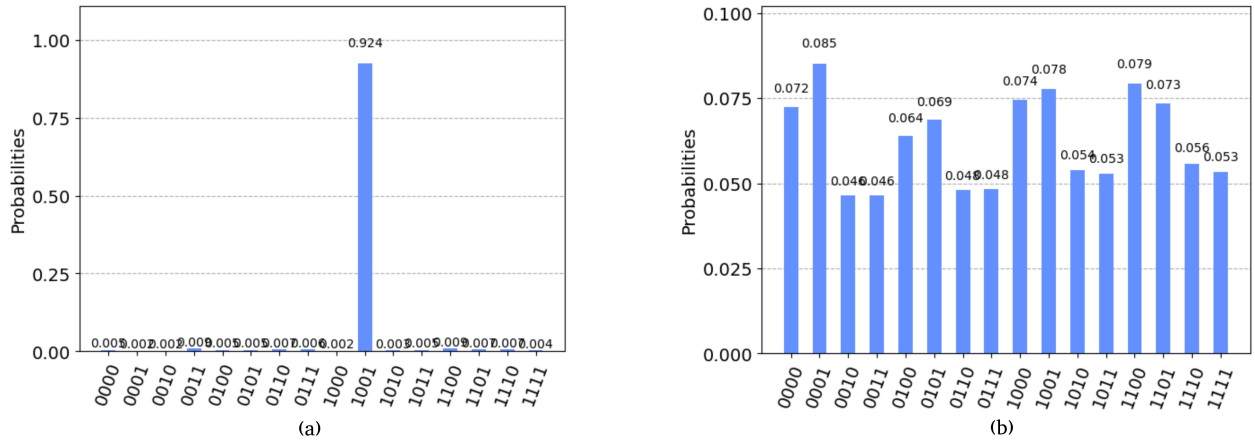


Figure 12: Iteration 2 : (a) Simulator : qasm_simulator (b) Real Quantum Device : ibmq_lima

3.1.3 Third Iteration

Probability of obtaining the correct target state 1001 in (a) Simulator = 0.960 and (b) Real Device = 0.071.

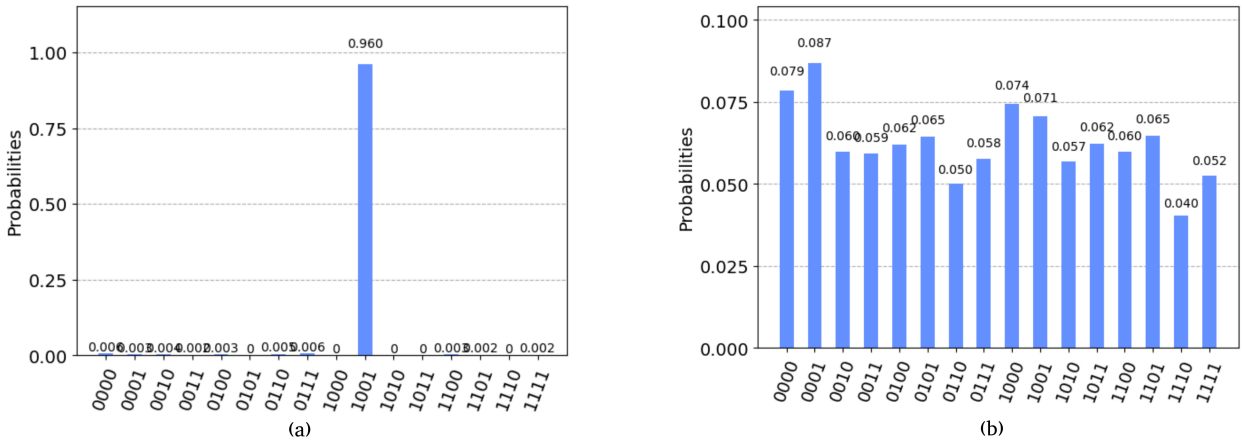


Figure 13: Iteration 3 : (a) Simulator : qasm_simulator (b) Real Quantum Device : ibmq_lima

3.2 Observations

1. **Simulator (qasm_simulator)** : The probability of obtaining the target state 1001 improves with every iteration - (0.490, 0.924, 0.960). This is the expected result as with every iteration, the state vector rotates closer to the target state in the N-dimensional Hilbert space, thereby increase its component along $|t\rangle$. The probability of all other states is very small and decreases further with each iteration. For N=16, after 3 iterations the probability of the target state is very high and that of others is negligible. The values of the probability vary a little from the expected theoretical values due to error in the system.

2. **Real Device (imbq_lima)** : After the first iteration the correct target state has the highest probability, though low compared to the expected number, but the results after the second and third iteration are far from the desired outcome. The probability distribution appears quite random and the chances of reading the target state on measurement are not improved by the algorithm since all states have probabilities close to each other.

3. **Comparison** The simulation results are clearly far superior to those of the algorithm performed in the real system.

3.3 Inferences

The implementation of the algorithm on the simulator has produced successful results. The results of the real device indicate that the current hardware is not yet fit for such computation due to the excessively high errors. These flaws result from decoherence and gate errors. Contrary to theory, higher iterations have produces larger errors due to the increase in the number of imperfect gates. There is need to improve the current hardware to make a fault tolerant system.

4 Additional notes and explanations

The **unstructured database search** involves finding a particular item from a given list/database of many items. The order or structure of this database, if it exists, is unknown leading to a wider applicability of the search. Knowledge about the structure of the database can reduce the complexity of the problem and provide higher speed-up.

4.1 Classical database search

Statement If the success probability of a single trial is p, the average number of trials required to achieve success is $\frac{1}{p}$.

Proof Let the random variable X be the number of trials required for success.
The probability of success in the n^{th} trial exactly = Probability of failure in the first n-1 trial and success in the n^{th} trial.

$$P(X = n) = (1-p)^{n-1}p$$

The average number of trials required = $\sum_{X=1}^{\infty} XP(X)$

$$N_{trials} = p + 2(1-p)p + 3(1-p)^2p + ... + n(1-p)^{n-1}p + ...$$

$$N_{trials} = \sum_{n=1}^{\infty} n(1-p)^{n-1}p$$

$$N_{trials} = p(\sum_{n=1}^{\infty}(1-p)^{n-1} + \sum_{n=2}^{\infty}(1-p)^{n-1} + \sum_{n=3}^{\infty}(1-p)^{n-1} + ...)$$

$$N_{trials} = p\left(\frac{1}{1-(1-p)} + \frac{1-p}{1-\frac{1}{1-p}} + \frac{(1-p)^2}{1-\frac{1}{1-p}} + ...\right)$$

$$N_{trials} = \frac{p}{1-1+p}\left(\sum_{n=0}^{\infty}(1-p)^n\right)$$

$$N_{trials} = \frac{1}{1-(1-p)} = \frac{1}{p}$$

For an unstructured database, the probability of finding the right item among a database of N items in a single trial is $\frac{1}{N}$ for a system without memory. Therefore, the average number of trials required for success is N. For a system with memory the average number of queries required reduces to $\lceil\frac{N+1}{2}\rceil$.

Another way of solving this problem is to first structure the database which requires an effort of the order $O(Nlog(N))$, followed by the search, which for a sorted database has a computational complexity of order $O(log(N))$.

4.2 The Quantum Algorithm

The conceptualisation of the problem and the development of the solution can be studied as follows.

4.2.1 Mapping the classical database to the quantum Hilbert Space

N items can be labelled using log(N) bits. Perfectly distinguishable items are mapped to non overlapping states in the Hilbert space. A Hilbert space of N dimensions is required to represent N items where each item is labelled $|i\rangle$ such that the set $|i\rangle$, i = 1,2,..N forms an orthogonal basis.

The objective of the search is to find a particular state, called the target state henceforth, which is denoted by $|t\rangle$. $|t\rangle$ is basis vector, therefore $|\langle t|i\rangle|^2 = \delta_{ti}$

4.2.2 Initialisation and Problem Formulation

Initially, the target state is unknown and the initial state must have all the possibilities with equal probability. The initial state, say $|s\rangle$ has an equal component along all the basis vectors and $|\langle t|i\rangle|^2 = \frac{1}{N}$. $|s\rangle$ is an **unbiased, uniform superposition state**.

The problem is now defined in this framework as going from state $|s\rangle$ to $|t\rangle$.

4.2.3 Solution of the problem using the oracle and amplitude amplification

Geometric solution Consider the subspace spanned by $|s\rangle$ and $|t\rangle$, the $|s\rangle$-$|t\rangle$ plave. The vector $|s\rangle$ has to be rotated about the normal to this subspace to reach $|t\rangle$. The path hence taken by $|s\rangle$ is the shortest path or the **geodesic** between these two states. These features are the geometric properties of a metric space - a space in which metric or distance is well-defined and the distances obey the triangle inequality.
Let $d(a,b)$ be the distance between the points a and b. According to the triangle equality,

$$d(a,c) \leq d(a,b) + d(b,c)$$

This triangle inequality provides a means to obtain the shortest path in a metric space. The shortest path between $|s\rangle$ to $|t\rangle$ is a geodesic circle (grid circle) on the $|s\rangle$ -$|t\rangle$ plane.

Algebraic solution in the Hilbert space The state $|s\rangle$ can be created. Information about the state $|t\rangle$ is obtained by using an oracle. The **oracle** is a black box that can provide an answer to a binary question. The quantum advantage lies in the ability to ask a question to a state which is in superposition of many states. In the Hilbert space, the oracle is a unitary transformation and the answer is encoded as the multiplicative representation of $\mathbb{Z}_2$, that is $\{+1,-1\}$. The transformation multiplies the vector by -1 if it equals the target, else multiples it by $+1$.

$$U_{oracle}|s\rangle = U_{oracle}\left(a_t|t\rangle + \sum_{i,|i\rangle\neq|t\rangle} a_i|i\rangle\right)$$

$$U_{oracle}|s\rangle = \left(-a_t|t\rangle + \sum_{i,a_i|i\rangle\neq|t\rangle} a_i|i\rangle\right)$$

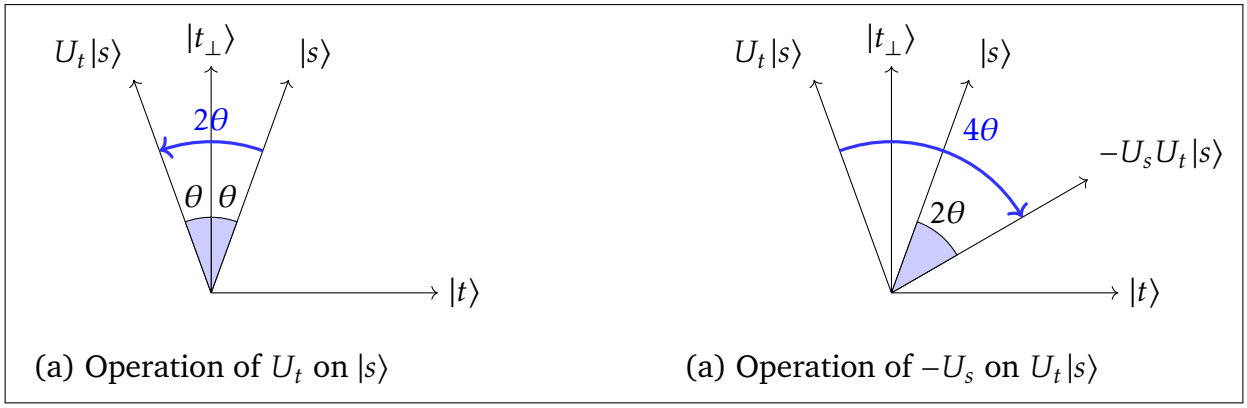
The oracle is implemented by the following combination of the identity and projection operator of $|t\rangle$:

$$U_t = I - 2P_t = I - 2|t\rangle\langle t|$$

Geometrically, it produces a reflection about the direction orthogonal to $|t\rangle$, say $|t_\perp\rangle$. Another such reflection operator that can be created is the reflection about the direction orthogonal to $|s\rangle$.

$$U_s = I - 2P_s = I - |s\rangle\langle s|$$

The operator $-U_s$ produces a reflection about $|s\rangle$.



(a) Operation of U_t on $|s\rangle$ (a) Operation of $-U_s$ on $U_t|s\rangle$

Figure 14: Pictorial representation of the reflection operations in an iteration of Grover's algorithm

Grover's algorithm involves applying the operations U_t and $-U_s$ alternatively to take the state from $|s\rangle$ to $|t\rangle$.

$$(-U_S U_t)^Q|s\rangle = |t\rangle, \quad \text{where Q is the number of iterations/queries}$$

With sufficient number of questions, we can obtain the state $|t\rangle$ from $|s\rangle$.

The angle between $|s\rangle$ and $|s = t\rangle$ is given by the dot product of the vectors.

$$cos(\frac{\pi}{2} - \theta) = \langle s|t\rangle = \left|\frac{1}{\sqrt{N}}\right| \implies sin(\theta) = \left|\frac{1}{\sqrt{N}}\right| \implies \theta = sin^{-1}\left(\frac{1}{\sqrt{N}}\right)$$

With each iteration the vector is rotated by an angle of 2θ away from $|s\rangle$ towards $|t\rangle$. After Q iterations, the vector is at angle $(2Q)\theta$ from $|s\rangle$ and $(2Q+1)\theta$ from $|t\rangle$. The solution is reached when the vector reaches sufficiently close (in the neighbourhood of upto angle θ) to $|t\rangle$, i.e., $(2Q+1)\theta \approx \frac{\pi}{2}$.

$$Q = \frac{1}{2}\left(\frac{\pi}{2sin^{-1}\left(\frac{1}{\sqrt{N}}\right)} - 1\right)$$

The number of iterations needed to find an item in a database of size N is the number of queries to the oracle and is called the **oracle complexity** of the problem.

For Q=1, N=4. This means that by asking a binary question/ query, 4 distinct possibilities can be identified, whereas in a classical algorithm only 2 items can be distinguished. When the database is large ($N \to \infty$), asymptotically,

$$sin\left(\frac{1}{\sqrt{N}}\right) = \frac{1}{\sqrt{N}} \implies Q \approx \frac{\pi}{4}\sqrt{N}$$

The best probabilistic result using quantum machinery is $Q \approx 0.57\sqrt{N}$.

If the state is an angle α from $|t\rangle$ such that $\alpha < \theta$ (which is guaranteed since the vector rotates in the direction from $|s\rangle$ to $|t\rangle$ in every iteration), then the error can be given as the dot product of the vectors, $cos(\alpha)$. The probability of success is hence $cos^2(\alpha)$.

$$\alpha < \theta \implies cos^2(\alpha) > cos^2(\theta)$$

$$cos^2(\theta) = 1 - sin^2(\theta) = 1 - \frac{1}{N} \implies cos^2(\alpha) = 1 - \frac{1}{N}$$

The algorithm will succeed with a probability that is at least $1 - \frac{1}{N}$. For a large N, the success probability can get very close to 1, though there is always a small chance of failure. This is remedied by performing multiple trials.

4.3 Grover's Algorithm using Wave Dynamics

The essence of Grover's Algorithm lies in the amplitude amplification of a required item using a clever interference. This interference experiment can be performed using only waves/oscillatory dynamics also, if there is wave mode corresponding to each basis vector (item in the database). The interference corresponds to the phenomena of beats in wave dynamics. In a system of oscillators, amplitude of one oscillator can be transferred to another by appropriate coupling between the oscillators.

5 References

1. Michael A. Nielsen and Isaac L. Chuang. 2011. Quantum Computation and Quantum Information: 10th Anniversary Edition (10th. ed.). Cambridge University Press, USA.

2. Phillip Kaye, Raymond Laflamme, and Michele Mosca. 2006. An Introduction to Quantum Computing. Oxford University Press.

3. Vera Blomvist Karlsson, Philip Stromberg, '4-qubit Grover's algorithm implemented for the ibmqx5 architecture', Degree project in computer science, School of Electrical Engineering and Computer Science, KTH Royal Institute Of Technology.

4. 'Grover's Algorithm', Qiskit Textbook: https://qiskit.org/textbook/ch-algorithms/grover.html

5. Lecture Notes by Dr. Apoorva Patel.

6 Appendix A : Code

```
#import packages
import qiskit
import math
from qiskit import *
from qiskit import IBMQ
#save account
IBMQ.save_account('dcfe6e2e0e82b73baf327399f9ade83a4e5cd910b87a6b642c5611
f5a3d06b0076f9ccf35cfc6ad4d50e7ce3e57321373bdaab5ceea4ccc2dd05770d094279a9')

# Step 1: Initialisation of the registers
#Creating the reqisters (4 quantum and 4 classical) and creating the circuit
q = QuantumRegister(4,'q')
c = ClassicalRegister(4,'c')
grover4 = QuantumCircuit(q,c)
#Using Hadamard gate H to put all states in superposition
grover4.h(q)
#circuit diagram
matplotlib inline
grover4.draw()

#setting the loop for iterations
for i in range(3):

    #Step 2: Making the Oracle Ut for 1001
    grover4.x(q[1])
    grover4.x(q[2])
    pi =math.pi
    grover4.cu1(pi/4, q[0], q[3])
    grover4.cx(q[0], q[1])
    grover4.cu1(-pi/4, q[1], q[3])
    grover4.cx(q[0], q[1])
    grover4.cu1(pi/4, q[1], q[3])
    grover4.cx(q[1], q[2])
    grover4.cu1(-pi/4, q[2], q[3])
    grover4.cx(q[0], q[2])
    grover4.cu1(pi/4, q[2], q[3])
    grover4.cx(q[1], q[2])
    grover4.cu1(-pi/4, q[2], q[3])
    grover4.cx(q[0], q[2])
    grover4.cu1(pi/4, q[2], q[3])
    grover4.x(q[1])
    grover4.x(q[2])

    #Step 3: The Diffuser (Amplification)
```

```
    #Hadamard gate H on all qubits
    grover4.h(q)
    grover4.x(q)
    grover4.cu1(pi/4, q[0], q[3])
    grover4.cx(q[0], q[1])
    grover4.cu1(-pi/4, q[1], q[3])
    grover4.cx(q[0], q[1])
    grover4.cu1(pi/4, q[1], q[3])
    grover4.cx(q[1], q[2])
    grover4.cu1(-pi/4, q[2], q[3])
    grover4.cx(q[0], q[2])
    grover4.cu1(pi/4, q[2], q[3])
    grover4.cx(q[1], q[2])
    grover4.cu1(-pi/4, q[2], q[3])
    grover4.cx(q[0], q[2])
    grover4.cu1(pi/4, q[2], q[3])
    grover4.x(q)
    grover4.h(q)

#Step 4: Measuring the qubits
grover4.barrier(q)
grover4.measure(q[0], c[0])
grover4.measure(q[1], c[1])
grover4.measure(q[2], c[2])
grover4.measure(q[3], c[3])
grover4.draw()

#Simulation
simulator = Aer.get_backend('qasm_simulator')
result = execute(grover4,backend = simulator).result()
#Simulation Results
from qiskit.tools.visualization import plot_histogram
plot_histogram(result.get_counts(grover4))

#load account and implement on real device
IBMQ.load_account()
provider = IBMQ.get_provider(hub='ibm-q', group='open', project='main')
qcomp = provider.get_backend('ibmq_lima')
job = execute(grover4,backend=qcomp)
from qiskit.tools.monitor import job_monitor
job_monitor(job)
result = job.result()
plot_histogram(result.get_counts(grover4))
```

7 Appendix B : Oracle implementation for various target states

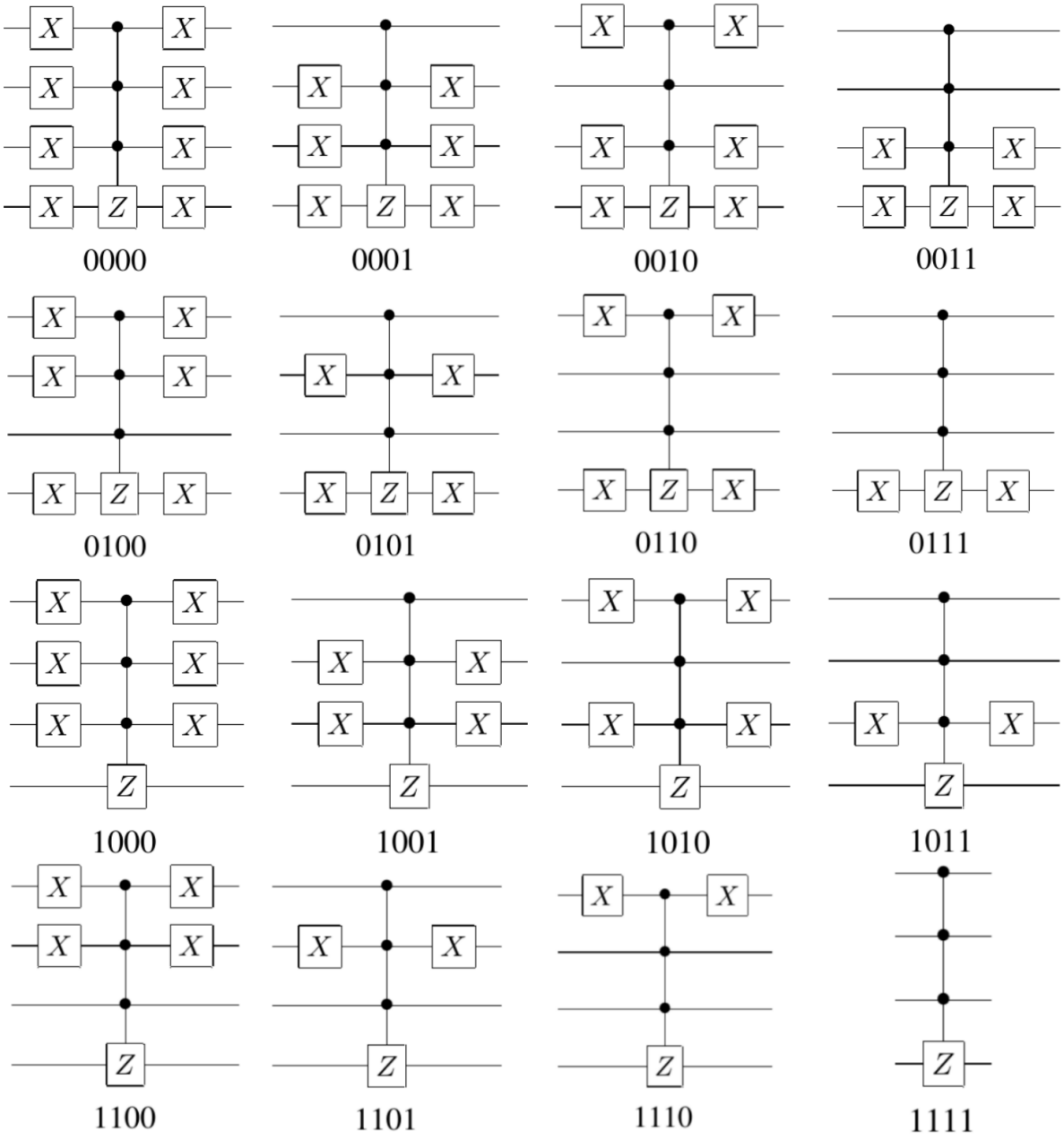


Figure 15: Gate implementation of oracles for N = 16