GENERALIZATIONS OF HIDDEN SUBGROUP ALGORITHMS

by

Damla Poslu

B.S. in Computer Engineering, Boğaziçi University, 2003

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2005

ACKNOWLEDGEMENTS

First of all, I would like to thank Prof. Cem Say who is the supervisor of this thesis. It is hard for me express my appreciation and admire for his never-ending patience, his will to look for the best and his generosity to share his knowledge. Whenever I lost my hope, his advices and his guidance light my way.

I also want to thank Prof. Fahrünisa Neyzi who provide me the details of Quantum Mechanics in the best way I have ever experienced. Her clear, detailed and warm approach not only to Quantum Mechanics but also people and life became a great inspiration to me.

My special thanks go to Prof. Levent Akın for his willingness on this thesis and his guidance on computation starting with Data Structures from my junior years. I also would like to thank Prof. Yaşar Safkan for his quantum computation classes which always force us to think differently.

I want to thank my sister Ezel for being my best friend. Her efforts to make me someone better will always help me.

I want to thank my fiancee Sait for his patience and help to finish this thesis.

Last of all I would like to thank my parents. Without their support, love and concern anything would be impossible.

ABSTRACT

GENERALIZATIONS OF HIDDEN SUBGROUP ALGORITHMS

In the future, it can be possible to store bit information in atoms. In that case, classical mechanics will not be enough to explain the atomic level model. Instead quantum mechanics will have to be used. A quantum bit exists as a superposition of 0 and 1. Creating superpositions and making parallel computation on them will allow faster solutions than classical computation. The field of quantum computation examines the possibility of using these physical properties for solving computational properties more efficiently.

In this thesis, we consider the problem of generalizing some quantum algorithms so that they will work on input domains whose cardinality is not necessarily powers of two. When analyzing the algorithms we assume that generating superpositions of arbitrary subsets of basis states whose cardinalities are not necessarily powers of two perfectly is possible. We have taken Ballhysa's model as a template and have extended it to Chi, Kim and Lee's generalization of the Deutsch-Jozsa algorithm and to Simon's algorithm.

ÖZET

GİZLİ ALTGRUP ALGORİTMALARININ GENELLEMELERİ

Gelecekte bir bitlik bilginin bir atomla gösterilmesi söz konusu olacaktır. Bu durumda, klasik mekanik atomik seviyede incelenen bu modeli açıklamada yetersiz kalacaktır. Bunun yerine, kuantum mekaniği kullanılmak zorunda kalınacaktır. Kuantum biti, 0 ve 1'in süperpozisyonları şeklinde var olmaktadır. Bu süperpozisyonların yaratılabilmesi ve üzerlerinde paralel hesaplama yapılabilmesi klasik hesaplamadan daha hızlı çözümlere izin vermektedir. Kuantum hesaplaması, bu fiziksel özelliklerden yararlanılarak bilişim problemlerinin nasıl daha verimli şekilde çözülebileceğini inceleyen araştırma alanıdır.

Bu tezde, bazı kuantum algorithmalarının genelleştirilmesi problemi incelenmiştir, böylece bu algoritmalar tanım kümesinin eleman sayısının ikinin üssü olmak zorunda olmadığı durumlarda çalışacaklardır . Bu algoritmaları analiz ederken, eleman sayılarının ikinin üssü olması gerekmeyen bazı vektör kümelerinin eşit olasılıklı süperpozisyonlarının mükemmel üretilmesinin mümkün olduğu varsayımında bulunulmuştur. Ballhysa'nın öngördüğü model örnek kabul edilmiş, Chi, Kim ve Lee'nin Deutsch-Jozsa algoritması genellemesi olarak hazırladıkları algoritmalar ile Simon'ın algoritmasına uyarlanmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

H	Hadamard Matrix
$\mathcal{H}$	Hilbert Space
I	Identity Matrix
U	Unitary Operator, Unitary Matrix
Z	Integers
$\imath$	$\sqrt{-1}$
σ_z	Pauli-Z= Phase-flip Operator
$\otimes$	Kronecker Product, Tensor Product
$\oplus$	XOR Operation in Boolean Algebra
$\lceil x \rceil$	Ceiling Function
CKL	Chi, Kim and Lee Algorithm
ESG	Equiprobable Superposition Generator
gcd	Greatest Common Divisor Algorithm
KSV	Kitaev, Shen and Vyalyi Algorithm

1. INTRODUCTION

In 1936 Alan Turing modelled a programmable computer which is called *Turing Machine*. The Turing Machine is supposed to have an infinite amount of secondary memory to read data, make the computation and write the output. There is also the concept of *Universal Turing Machine* which is proven to be able to simulate any other Turing Machine, meaning that if an algorithm is performed on any piece of hardware, there exists an equivalent algorithm for the Universal Turing Machine to complete the same task. The Universal Turing Machine became the mathematical concept to define the class of algorithms to be implemented on a hardware [1].

However, this model is classical. There exist problems that can not be solved efficiently in polynomial time complexity. Scientists as Feynman realized that another model of computation, which relies on the rules of quantum mechanics instead of the classical ones, may be used to solve problems. The main difference in this model was that quantum mechanics can describe the states in atomic level which reflects itself in the basic representation of a single bit. In classical computation, our bits are either 0 or 1. However, in quantum computation, a bit can be in both states, each of them up to a certain degree.

In 1985, Deutsch presented the first quantum algorithm [2], which is superior to the best classical algorithm for the same problem. This was not an exponential superiority, however it became an important step towards constructing quantum algorithms. He also showed that the Fourier transformation over the group Z_2^n could be implemented efficiently, which we call as Hadamard transformation. Bernstein and Vazirani showed the existence of a *Universal Quantum Turing Machine* which could simulate any quantum computer with polynomial slowdown [3]. In 1992, Deutsch and Jozsa [4] generalized Deutsch's algorithm for n-bit strings and came up with an algorithm which brings exponential speedup over the best classical algorithm for the same problem.

In 1994, Simon [5] presented a problem with its quantum algorithm of polynomial time, which has no polynomial solution classically. This algorithm became a basis for the most fascinating and important algorithm that attracts attention to the quantum computation world. It was presented by Shor in 1994. He introduced polynomial time algorithms for factoring an integer and extracting the discrete logarithm [6].

There are other important algorithms such as the search algorithm proposed by Grover in 1996 [7] which does not bring an exponential speedup, however do their job more efficiently than the best classical search algorithms. There is also the quantum random walk graph traversal algorithm, whose superiority above classical methods was demonstrated by Childs et al. in 2002 [8].

The publication of Shor's algorithm caused a great deal of excitement in the field of quantum computing. It has been eleven years since Shor's algorithm has been presented, however since then no comparable advance has been made in the field of computation. This may be due to the fact that it is hard to define a problem that may have a quantum solution which is more efficient than the existing classical solution [1, 9] or due to another fact that we are used to think classically as stated in [1] such that it becomes hard to adapt to the conceptual difficulties and differences of quantum theory.

Besides, the physical implementation of quantum algorithms has a long way to go, since the most advanced quantum computer is a 7-bit one to factor the number 15. So the implementation of quantum algorithms is also a challenging task.

In this thesis, we provide algorithms which generalizes two algorithms by Chi, Kim and Lee and Simon's algorithm to work on domains whose cardinalities are not necessarily powers of two, assuming that we have an operator to produce equiprobable superpositions of the elements of a given set S.

In the remainder of this thesis, we will cover the fundamentals of quantum computing in Chapter 2. In chapter 3, the circuit notation of quantum computation, with

gates corresponding to instructions will be discussed. In chapter 4, a detailed discussion of basic quantum algorithms, namely the Deutsch-Jozsa algorithm, Simon's algorithm and Shor's factorization algorithm will be given. In the same chapter several different generalized versions for the Deutsch-Jozsa algorithm, namely Chi, Kim and Lee's generalizations for evenly-distributed and evenly-balanced functions and Ballhysa's generalization for arbitrary domains will be discussed. In Chapter 5, we will discuss alternative algorithms for Chi, Kim and Lee's generalizations and Simon's algorithm on arbitrary domains. Chapter 6 will be the conclusion.

2. FUNDAMENTALS OF QUANTUM COMPUTATION

2.1. Quantum Bits

The essential difference between quantum and classical computation comes into play in the definition of the quantum bit. In quantum computation, a bit does not have to be in one of the states 0 or 1, and it can be in a combination of the states 0 and 1, as stated in [1]. Instead of saying quantum bits, we refer to them simply as qubits. Furthermore, in order to emphasize the difference, we use the Dirac notation, $|\ \rangle$, of the physics world. As a result, the states $|0\rangle$ and $|1\rangle$ form the orthonormal basis for the vector space of the qubits, which we call *Hilbert Space*, $\mathcal{H}$, and states in this space can be defined as *linear combinations* , generally called *superpositions* , of these basis states as follows:

$$|\psi\rangle \quad = \quad \alpha|0\rangle + \beta|1\rangle$$

The numbers α and β are complex numbers,denoting the amplitudes of the basis states of a given state. The state of a qubit can be viewed as a vector in $\mathcal{H}$ such that the state $|0\rangle$ is represented as $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and the state $|1\rangle$ as $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$. As a result, another way of representing the state $|\psi\rangle$ would be:

$$\begin{aligned} |\psi\rangle &= \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \beta \end{bmatrix} \\ &= \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \end{aligned}$$

Since a given state is normalized in the vector space:

$$\langle\psi|\psi\rangle = 1 \tag{2.1}$$

As mentioned in the previous paragraph, states $|0\rangle$ and $|1\rangle$ form orthonormal basis states and are orthogonal to each other, resulting in $\langle 0|1\rangle = 0$, $\langle 0|0\rangle = 1$ and $\langle 1|1\rangle = 1$. Therefore

$$\begin{aligned} \langle\psi|\psi\rangle &= 1 \\ (\alpha^*\langle 0| + \beta^*\langle 1|)(\alpha|0\rangle + \beta|1\rangle) &= \alpha^*\alpha\langle 0|0\rangle + \alpha^*\beta\langle 0|1\rangle + \beta^*\alpha\langle 1|0\rangle + \beta^*\beta\langle 1|1\rangle \\ |\alpha|^2 + |\beta|^2 &= 1 \end{aligned}$$

As a result, when we discuss the contents of a qubit, instead of saying definitely one of the values, we will say that the value which will appear upon measurement is $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. It is also easy to see that the probability values correspond to the squares of the amplitudes of the states. Clearly, the number of different amplitudes that the states can have is uncountable. This property constitutes one of the basic differences between quantum computation and classical computation.Instead of a pure state of $|0\rangle$ or $|1\rangle$, there are uncountably many mixed states of both.

Another method to explain the qubits and their states is using the geometric representation. It is the *Bloch Sphere* [1] consisting of the unit vectors in the three dimensional space and it is accepted that any normalized state can be expressed geometrically in this sphere. Each and every state is a point in the *Bloch Sphere* and described by the angles θ and φ. As seen in Figure 2.1, the components of the state ψ are $\cos\theta$ in the z direction, $\sin\theta\cos\varphi$ in the y direction and $\sin\theta\sin\varphi$ in the x direction.

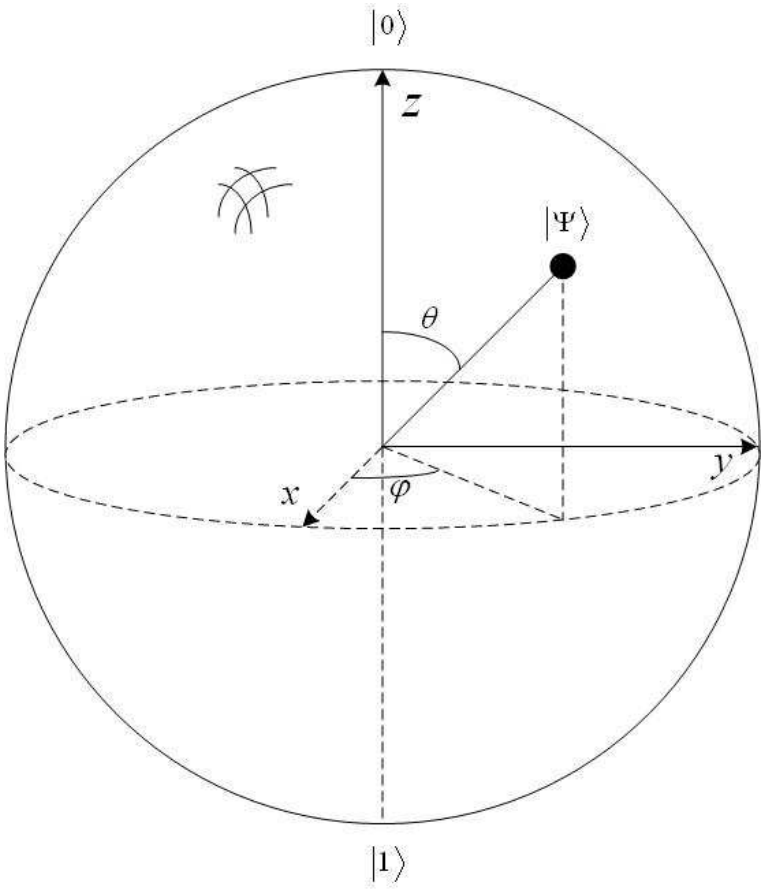

Figure 2.1. Bloch Sphere representation for single qubit system

Using these components, any state can be defined as shown in [10]:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle \tag{2.2}$$

If we want to analyze whether the above notation is normalized or not:

$$\begin{aligned}
\langle\psi|\psi\rangle &= (\cos\frac{\theta}{2}\langle 0| + e^{-i\varphi}\sin\frac{\theta}{2}\langle 1|)(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle) \\
&= \cos^2\frac{\theta}{2}\langle 0|0\rangle + e^{-i\varphi}e^{i\varphi}\sin^2\frac{\theta}{2}\langle 1|1\rangle \\
&= \cos^2\frac{\theta}{2} + \sin^2\frac{\theta}{2} \\
&= 1
\end{aligned}$$

As seen in the above equation, the normalization constraint in equation 2.1 is satisfied. It is an unfortunate fact that we can specify only single qubits on the Bloch Sphere.

2.2. Multiple Qubits

In classical computation, a register holds n bits and it is possible to express 2^n numbers in that register. In quantum computation we use the tensor product $\otimes$ to form the state space of a register from the state spaces of its individual qubits [1]. The tensor product of the two matrices A and B is given as:

$$\begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix} \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \dots & a_{nm}B \end{pmatrix}$$

The same logic is valid in the tensor product of the two vectors:

$$\begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = [a_1b_1 \dots a_1b_m a_2b_1 \dots a_2b_m \dots a_nb_1 \dots a_nb_m]^T$$

If we consider a system with two qubits, then the state can be represented as the combination of the states $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$ with amplitudes $\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}$ respectively.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

We can say that if we make a measurement on the state $|\psi\rangle$ we will observe with probability $|\alpha_{00}|^2$ the value $|00\rangle$ and with probability $|\alpha_{01}|^2$ the value $|01\rangle$, with

probability $|\alpha_{10}|^2$ the value $|01\rangle$ and with probability $|\alpha_{11}|^2$ the value $|11\rangle$. Since the normalization constraint is valid through all states in quantum computation, the sum of the probabilities is equal to 1. Instead of using the binary notation, if we use decimal notation our state becomes:

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle + \alpha_3|3\rangle$$

In short

$$|\psi\rangle = \sum_{t=0}^{3} \alpha_t|t\rangle \tag{2.3}$$

The normalization constraint can be defined as:

$$\sum_{t=0}^{3} |\alpha_t|^2 = 1 \tag{2.4}$$

As seen in the above example, every vector $|i\rangle$ denoted in the decimal form is a vector consisting of all 0's except for a 1 in the $(i+1)^{th}$ column. For example the state $|3\rangle$ is expressed as $[0\ 0\ 0\ 1]^T$. As a result, the equations in 2.3 and 2.4 can be generalized as:

$$|\psi\rangle = \sum_{t=0}^{2^n-1} \alpha_t|t\rangle$$

$$\sum_{t=0}^{2^n-1} |\alpha_t|^2 = 1$$

3. QUANTUM COMPUTATION

A circuit notation, with gates corresponding to instructions, turns out to be convenient to describe quantum algorithms.

3.1. Single Qubit Gates

The basic idea lying behind the gates is to transform a certain state to another one. If we want to express this geometrically, in section 2.1 we mentioned that a state of a single qubit is a point on the surface of the Bloch Sphere. As a result, a certain gate will carry the current point of the state to another point on the surface of the sphere. Any gate making this transformation can be represented with a matrix mathematically. If we express this matrix with the letter U as the operator applied on the state then in order to carry a unit state to another, the operator must be unitary.

$$UU^\dagger = I \tag{3.1}$$

where $U^\dagger$ is defined to be the conjugate transpose [1] of the operator. It can be shown as:

$$U^\dagger = (U^*)^T$$

In order to find out the effect of a unitary operator U on a state defined as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, it is enough to know the effect of U on the states $|0\rangle$ and $|1\rangle$ individually. We can say that if

$$\begin{aligned} U|0\rangle &= a|0\rangle + b|1\rangle \\ U|1\rangle &= c|0\rangle + d|1\rangle \end{aligned}$$

it is easy to find out the matrix representation of the unitary operator U as

$$U = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

with the constraint in equation 3.1. If we also want to show *dagger* of the operator:

$$\begin{aligned} U^\dagger &= \left(\begin{bmatrix} a & c \\ b & d \end{bmatrix}^* \right)^T \\ &= \left(\begin{bmatrix} a^* & c^* \\ b^* & d^* \end{bmatrix} \right)^T \\ &= \begin{bmatrix} a^* & b^* \\ c^* & d^* \end{bmatrix} \end{aligned}$$

The effect of the operator on an arbitrary state $|\psi\rangle$ is:

$$\begin{aligned} U|\psi\rangle &= U(\alpha|0\rangle + \beta|1\rangle) \\ &= \alpha U|0\rangle + \beta U|1\rangle \\ &= \alpha(a|0\rangle + b|1\rangle) + \beta(c|0\rangle + d|1\rangle) \\ &= (\alpha a + \beta c)|0\rangle + (\alpha b + \beta d|1\rangle) \end{aligned}$$

Another way of showing this equality is using the matrix method:

$$\begin{aligned} U|\psi\rangle &= \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\ &= \begin{bmatrix} a\alpha + c\beta \\ b\alpha + d\beta \end{bmatrix} \end{aligned}$$

With this knowledge about the unitary operators, we can try to describe the gates

which will carry out some special transformations. These gates other than known classical gates have a very crucial property for the reversibility of the evolution in quantum physics [10]. They are reversible gates in the sense that when we look at the output of any of them we are able to understand what the input is.

The first one of them is the NOT gate which transforms the state $|0\rangle$ to the state $|1\rangle$ and vice versa.

$$\begin{aligned} U_{NOT}|0\rangle &= |1\rangle \\ U_{NOT}|0\rangle &= |0\rangle \end{aligned}$$

So by elementary operations it is easy to see that the matrix representation we are looking for is:

$$U_{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The NOT gate is one of the Pauli spin matrices [1] constituting the basis vectors in a 3-dimensional complex numbers world with the identity matrix. The NOT gate corresponds to σ_x of the Pauli spin matrices whereas the identity matrix corresponds to σ_0. There are also σ_y and σ_z [11] defined as:

$$\sigma_0 \equiv I \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad , \quad \sigma_1 \equiv \sigma_x \equiv X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
$$\sigma_2 \equiv \sigma y \equiv Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad , \quad \sigma_3 \equiv \sigma z \equiv Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

As mentioned before, all these operators are unitary. The Z gate is also very important in the sense that it negates the state if the state is $|1\rangle$, and has no effect

otherwise. This gate will play an important part in CKL (Chi, Kim and Lee Algorithm) of section 4.1.

Since Pauli spin matrices form basis matrices of the Hilbert space in 2×2 dimensional space, we can easily say that any state in 2×2 space can be represented as the combination of the Pauli matrices.

Another very essential gate is the Hadamard gate which we will use very frequently. The effect of the Hadamard gate to the basis states:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{3.2}$$

$$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{3.3}$$

Therefore the matrix representation of the Hadamard gate is :

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

The basic aspect of the Hadamard gate is that when given a state of consisting all 0's and apply Hadamard transformation, we get the superposition of all possible states that can be described by n qubits, with the same amplitudes and so the same probabilities. When we consider the effect of the Hadamard gate on the Bloch Sphere we see that the gate applies a 90° rotation by the y axis first then a rotation by the x axis of 180° [1]. As we will see with the Hadamard gate, if we apply it twice to any state, the state will remain unchanged. Hadamard gate which is a reversible gate has also another property as:

$$H^2 = I$$

The inverse of the Hadamard matrix is itself. Such operators are called self-inverse operators.

There are two important 2×2 operators called,the phase gate, S, and $\pi/8$ gate, T, respectively. The matrix representations of the operators are:

$$\begin{aligned} S &= \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \\ T &= \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} \end{aligned}$$

Another interesting issue is that when we exponentiate the Pauli matrices we observe rotation matrices $R(\theta)$ around the corresponding axis.

$$\begin{aligned} R_x(\theta) &= e^{-iX\frac{\theta}{2}} \\ &= \cos(\frac{\theta}{2})I - i\sin(\frac{\theta}{2})X \\ &= \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} R_y(\theta) &= e^{-iY\frac{\theta}{2}} \\ &= \cos(\frac{\theta}{2})I - i\sin(\frac{\theta}{2})Y \\ &= \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} R_z(\theta) &= e^{-iZ\frac{\theta}{2}} \\ &= \cos(\frac{\theta}{2})I - i\sin(\frac{\theta}{2})Z \\ &= \begin{bmatrix} e^{\frac{-i\theta}{2}} & 0 \\ 0 & e^{\frac{i\theta}{2}} \end{bmatrix} \end{aligned}$$

There is a famous theorem [1] expressing that considering the Bloch Sphere and by making any kind of rotations around the y and z axis with any angle, we can come up with uncountable number of unitary operators. However, in practice it is not possible to implement all of them with zero error. Today it is quite clear that most of the quantum algorithms are not exact, instead bounded with some error.

Theorem 2.1 For any unitary operation U on a single qubit, there exist real numbers α, β, γ and δ such that

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta) \tag{3.4}$$

3.2. Multiple Qubit Gates

In multiple qubit operations, the issue is to put all the qubits under different kinds of gate transformations. When we consider a two qubit state as in Figure 3.1, the initial state is the tensor product of the states $|q_1\rangle$ and $|q_2\rangle$. As a result what comes out at the end of the transformations is the tensor product of the transformed states $U|q_1\rangle$ and $V|q_2\rangle$.

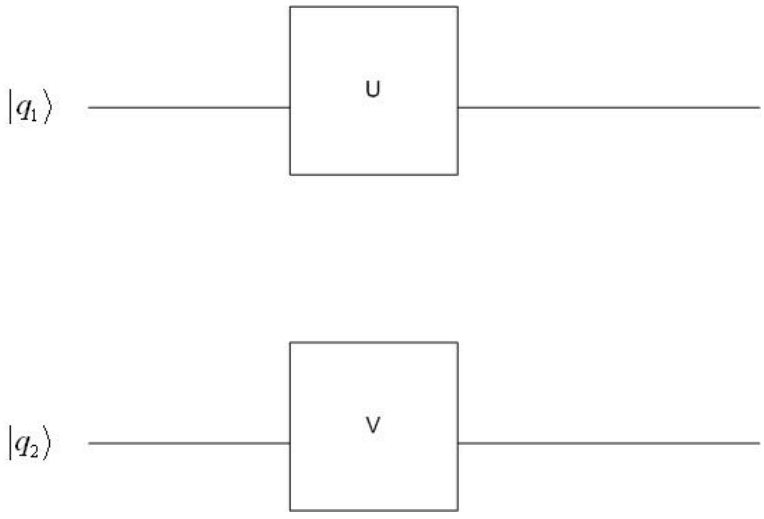

Figure 3.1. Representation of two unitary transformations on two different single qubits

Since we consider the tensor product both at the beginning and at the end, it is possible for us to express the overall effect of the two gates by taking their tensor product. Our new gate is the 4×4 matrix, $W = U \otimes V$.

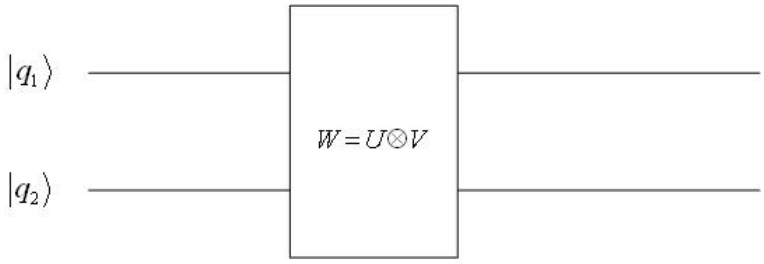

Figure 3.2. Overall effect of two unitary transformations on a two qubit system of Figure 3.1

If we consider the case that in a two qubit system, only one of the qubits is taken under the progress of a unitary transformation and the other one stays the same then we can consider that qubit as if it is under the effect of the identity matrix, I. As an example, the second qubit, $|q_2\rangle$ of the figure 3.1 may not be under any transformations then the gate V will apparently be the identity gate, I. Therefore the combination matrix W will be tensor product of U and I, as $W = U \otimes I$. The application two qubit example mentioned above can be generalized to an n qubit one with the usage of tensor product.

We can make controlled operations on the qubits, meaning that considering the state of one qubit we can decide to apply the operator U on the other bit. The most famous example of these operations is the CNOT gate which has the following effect:

$$\begin{aligned} CNOT|00\rangle &= |00\rangle \\ CNOT|01\rangle &= |01\rangle \\ CNOT|10\rangle &= |11\rangle \\ CNOT|11\rangle &= |10\rangle \end{aligned}$$

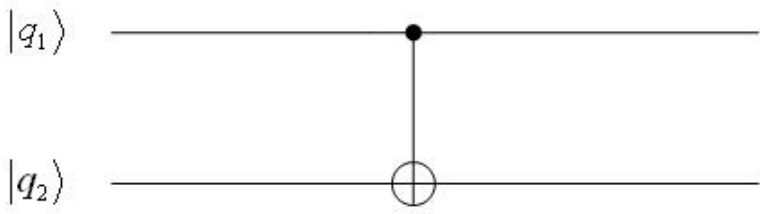

Figure 3.3. Circuit implementation of CNOT gate

Table 3.1. Truth table for CNOT gate

q_1	q_2	q_1'	q_2'
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

If we want to show the effect of the CNOT [1] gate mathematically, we can write it as $CNOT|xy\rangle = |x, x \oplus y\rangle$, where the effect of the first qubit is seen without changing it. Shortly, the CNOT gate transformation takes a look at the first qubit and if the first qubit is in state $|0\rangle$, it does nothing, if the state is $|1\rangle$ it acts as if a NOT gate is applied to the second qubit. Alternatively, we can define a zero-controlled gate where a 0 in the first qubit inverts the second qubit. Furthermore, the operation in the second qubit does not necessarily have to be the NOT operation, different unitary operators can be applied to the target bit.

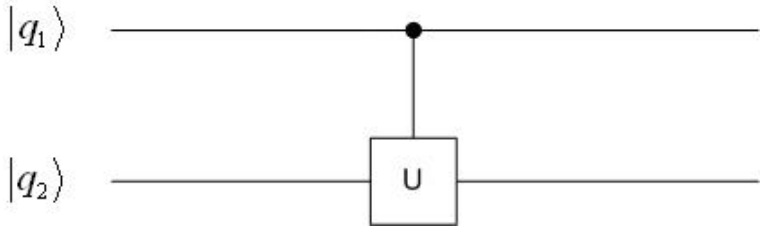

Figure 3.4. Circuit implementation of Controlled Unitary oepration

Table 3.2. Truth table for Toffoli gate

q_1	q_2	q_3	q_1'	q_2'	q_3'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

All of the discussions above were based on two qubit systems, however for a controlled operation the number of control qubits and target qubits can be more than one as well.

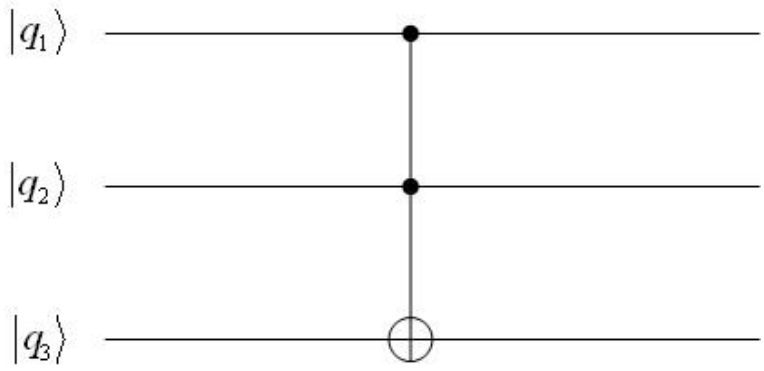

Figure 3.5. Circuit for Toffoli matrix

For example we can check for n qubits as the control qubits and reflect the change to m-qubit target qubits with $2^m \times 2^m$ unitary operator. One example is the Toffoli [10] gate which has the transformation $|x, y, z\rangle \rightarrow |x, y, z \oplus xy\rangle$ as in Figure 3.5. The truth table of Toffoli gate is shown in Table 3.2.

Another specific issue of the multiple qubit gates is the effect of the Hadamard

transformation. The effect of the Hadamard gate for single bit was:

$$\begin{aligned} H|0\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ H|1\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}} \end{aligned}$$

as we can remember from the equations 3.2 and 3.3. When we apply Hadamard transformation to all qubits of an n-qubit register, the result is:

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle \tag{3.5}$$

where $x \cdot y$ is defined as $x_1 y_1 + x_2 y_2 + ... + x_n y_n \ \ (mod\ 2)$ assuming that both x and y are n-bit numbers that can be represented as $x = \sum_{i=0}^{n-1} x_i 2^i$ and $y = \sum_{i=0}^{n-1} y_i 2^i$.

The equation 3.5 can be proved using the induction method. If the state $|x\rangle$ in equation 3.5 is $|0\rangle$, then the effect will be:

$$H^{\otimes n}|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} |y\rangle \tag{3.6}$$

It is obvious that the resulting state is a superposition of possible states between $|0\rangle$ and $|2^n - 1\rangle$ with the same probabilities.

3.3. Measurement

In all of the sections till this section, we talked about the probable states and their changes. However we never came up with an exact solution. The system is closed and everything depends on the probabilities. In order to break this up, we make measurements when we need. The basic issue of the measurement is that after the measurement, the state collapses into one of the possible states. The new existing state is the one measured. With a mathematical notation we can describe the measurement event as follows:

We have a generally defined measurement operator [1] which specifies itself according to the measured state as:

$$M_i = |i\rangle\langle i|$$

where i is the possible outcome of the measurement. And we express the probability of observing the state i after the measurement of the state $|\psi\rangle$ as:

$$p(i) = \langle\psi|M_i^\dagger M_i|\psi\rangle \tag{3.7}$$

If the outcome of the measurement is the state $|i\rangle$ then the old state $|\psi\rangle$ collapses to a new state $|\psi'\rangle$ as:

$$|\psi'\rangle = \frac{M_i|\psi\rangle}{\sqrt{p(i)}}$$

As an example, suppose that we have a state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. According to the definition, the measurement operators of the states $|0\rangle$ and $|1\rangle$ respectively are:

$$M_0 = |0\rangle\langle 0| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$
$$M_1 = |1\rangle\langle 1| = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

When we analyze the probabilities according to equation 3.7:

$$\begin{aligned} p(0) &= \langle\psi|M_0^\dagger M_0|\psi\rangle \\ &= [\alpha^* \quad \beta^*] \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \end{aligned}$$

$$
\begin{aligned}
&= [\alpha^* \quad \beta^*] \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\
&= [\alpha^* \quad \beta^*] \begin{bmatrix} \alpha \\ 0 \end{bmatrix} \\
&= |\alpha|^2
\end{aligned}
$$

$$
\begin{aligned}
p(1) &= \langle \psi | M_1^\dagger M_1 | \psi \rangle \\
&= [\alpha^* \quad \beta^*] \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\
&= [\alpha^* \quad \beta^*] \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\
&= [\alpha^* \quad \beta^*] \begin{bmatrix} 0 \\ \beta \end{bmatrix} \\
&= |\beta|^2
\end{aligned}
$$

And the transformed state for both of the possibilities, assuming that α and β are expressed as $a + bi$ and $c + di$ respectively:

$$
\begin{aligned}
|\psi_0'\rangle &= \frac{M_0 |\psi\rangle}{\sqrt{p(0)}} \\
&= \frac{\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}}{\sqrt{|\alpha|^2}} \\
&= \frac{\begin{bmatrix} a + bi \\ 0 \end{bmatrix}}{\sqrt{a^2 + b^2}} \\
&= \frac{a + bi}{\sqrt{a^2 + b^2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
&= \frac{a + bi}{\sqrt{a^2 + b^2}} |0\rangle
\end{aligned}
$$

When we check for the modulus of the amplitude of the state $|\psi'_0\rangle$:

$$\begin{aligned} |\frac{a+bi}{\sqrt{a^2+b^2}}| &= \frac{\sqrt{a^2+b^2}}{\sqrt{a^2+b^2}} \\ &= 1 \end{aligned}$$

So the observation of the state $|\psi'_0\rangle$ will yield the value $|0\rangle$ with probability 1.

With the same convention:

$$\begin{aligned} |\psi'_1\rangle &= \frac{M_1|\psi\rangle}{\sqrt{p_1}} \\ &= \frac{\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}}{\sqrt{|\beta|^2}} \\ &= \frac{\begin{bmatrix} 0 \\ \beta \end{bmatrix}}{|\beta|} \\ &= \frac{\beta}{|\beta|} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= \frac{\beta}{|\beta|}|1\rangle \end{aligned}$$

Clearly, the modulus of the amplitude of the state $|\psi'_1\rangle$ is 1, meaning $|1\rangle$ will be observed with probability 1.

On a more general perspective for a state $|\psi\rangle$ represented as $|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i|i\rangle$ the probability to observe the state $|i\rangle$ is:

$$p(i) = |\alpha_i|^2$$

and the new state upon measurement:

$$|\psi'\rangle \quad = \quad \frac{\alpha_i}{|\alpha_i|}|i\rangle$$

4. QUANTUM ALGORITHMS

4.1. Variants of the Deutsch-Jozsa Algorithm

One of the most important algorithms that give insight about quantum algorithms is Deutsch's algorithm [2, 12]. In Deutsch's problem, we have a function f and there are two possible options for f. It can be either constant, giving the same result no matter what the input is, or balanced, resulting in 0 for half of the possible inputs and 1 for the other half. There are two versions for it. In the first one, the size of the input is a single bit, either 0 or 1, meaning that our unknown function is of the form $f : \{0,1\} \rightarrow \{0,1\}$; the other version is the generalization of the first version of Deutsch's algorithm to the n qubit case by Deutsch and Jozsa [4]. The problem is to be able to understand whether the unknown function is constant or balanced. The question "If we do not know anything about the function how could we make decisions about it" may appear in minds, which takes us to our most important assumption in the algorithms. We assume that we have an oracle which can tell us the result of the function f for a given input. In subsections 4.1.1 and 4.1.2 both of the algorithms will be described in a detailed manner. Subsections 4.1.3 through 4.1.5 describe various generalizations of the Deutsch-Jozsa algorithm.

4.1.1. Deutsch's Algorithm

As mentioned in the introductory part and stated in [2, 12], we have a function $f : \{0,1\} \rightarrow \{0,1\}$. We know that there are two possibilities for this function; that it can be either constant or balanced. If it is constant then we can easily say that $f(0) = f(1)$ and the result is either 0 or 1. If it is balanced, then the definition of the function says that the function must return 0 for half of the inputs and 1 for the other half. Since we have two inputs, it is appropriate to say that the function f results differently for the input 0 and 1, shortly $f(0) \neq f(1)$. Besides we assume that we have an oracle, U_f which knows the results of the function for a given input and works as $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$.

If we try to compute whether the function f is constant or balanced, considering the single bit input case, with a classical computation we will need two oracle calls. We have to compute our function f for the input 0 and then for the input 1 and compare them. The computation will take two oracle calls. However with quantum computation we are able find out what we look for with one call of the function in the oracle. The algorithm is:

1. Initialize the first and second qubit to the states $|0\rangle$ and $|1\rangle$ respectively.
2. Apply Hadamard transformation to each qubit.
3. Apply the oracle U_f.
4. Apply Hadamard transformation once more to the state in the first qubit.
5. Measure the value of the first qubit.
6. If the value is zero then our function is constant, if the value is one then the function is balanced.

Figure 4.1. Quantum Algorithm for Deutsch's Problem

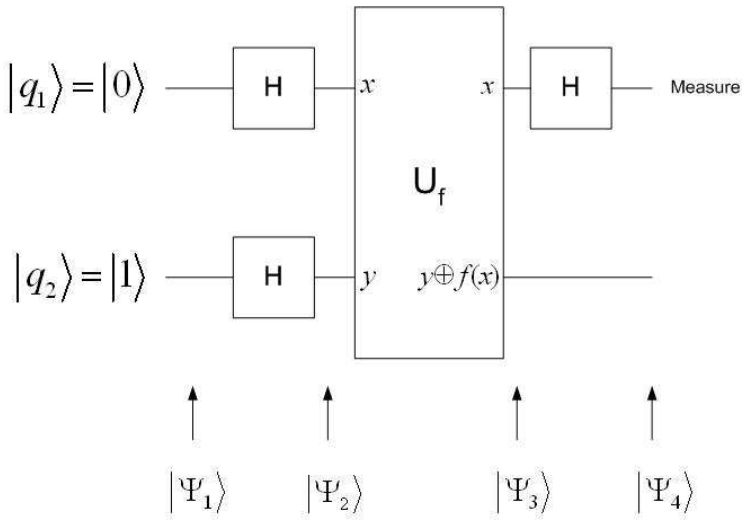

Figure 4.2. Circuit for Deutsch's algorithm

In order to look at how the algorithm works, we can explain it step by step as follows:

1- The initial state for the two qubit system is as follows:

$$|q_1\rangle = |0\rangle \quad , \quad |q_2\rangle = |1\rangle$$

As a result the total initial state of the system appears as:

$$|\Psi_1\rangle = |q_1\rangle \otimes |q_2\rangle = |0\rangle \otimes |1\rangle$$

2- Application of Hadamard transformation to each qubit results in:

$$\begin{aligned} |\Psi_2\rangle &= H \otimes H|\Psi_1\rangle = H|q_1\rangle \otimes H|q_2\rangle \\ |\Psi_2\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \end{aligned}$$

3- Application of the oracle U_f to the system where U_f is accepted to make the transformation $U_f|q_1, q_2\rangle = |q_1, q_2 \oplus f(q_1)\rangle$ without the explicit knowledge of the function f. So the oracle applied form of the system becomes:

$$|\Psi_3\rangle = U_f|\Psi_2\rangle = U_f\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

We know that we can rewrite $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ as $\frac{1}{\sqrt{2}}\sum_{x=0}^{1}|x\rangle$. We can interpret the above equation as below:

$$\begin{aligned} |\Psi_3\rangle &= U_f|\Psi_2\rangle = U_f\left(\frac{1}{\sqrt{2}}\sum_{x=0}^{1}|x\rangle\right) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\ &= \frac{1}{2}\left(\sum_{x=0}^{1}|x\rangle\right) \otimes ((|0 \oplus f(x)\rangle) - (|1 \oplus f(x)\rangle)) \end{aligned}$$

We also know that:

$$\begin{aligned} |0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle &= \begin{cases} |0\rangle - |1\rangle, & \text{if } f(x) = 0; \\ |1\rangle - |0\rangle, & \text{if } f(x) = 1. \end{cases} \\ &= (-1)^{f(x)}(|0\rangle - |1\rangle) \end{aligned}$$

Therefore, $|\Psi_3\rangle$ becomes:

$$\begin{aligned}
|\Psi_3\rangle &= \frac{1}{2}\left(\sum_{x=0}^{1}|x\rangle\right)\otimes((|0\oplus f(x)\rangle)-(|1\oplus f(x)\rangle)) \\
&= \frac{1}{2}\left(\sum_{x=0}^{1}|x\rangle\right)\otimes(-1)^{f(x)}(|0\rangle-|1\rangle) \\
&= \frac{1}{2}\left(\sum_{x=0}^{1}(-1)^{f(x)}|x\rangle\right)\otimes(|0\rangle-|1\rangle) \\
&= \frac{(-1)^{f(0)}|0\rangle+(-1)^{f(1)}|1\rangle}{\sqrt{2}}\otimes\frac{(|0\rangle-|1\rangle)}{\sqrt{2}}
\end{aligned}$$

4- Applying Hadamard transform only to the first qubit yields:

$$\begin{aligned}
|\Psi_4\rangle &= (H\otimes I)|\Psi_3\rangle=\frac{1}{\sqrt{2}}H((-1)^{f(0)}|0\rangle+(-1)^{f(1)}|1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle) \\
&= \frac{1}{2}((-1)^{f(0)}(|0\rangle+|1\rangle)+(-1)^{f(1)}(|0\rangle-|1\rangle))\otimes\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)
\end{aligned}$$

If we analyze state $|\Psi_4\rangle$ for the cases where f is either constant or balanced, there are two options:

$$\begin{cases}
\text{If } f \text{ is constant}, & f(0)=f(1); \\
\text{If } f \text{ is balanced}, & f(0)\neq f(1).
\end{cases}$$

If f is constant, $f(0)=f(1)$, $f(0)=f(1)$ can be both 0 or 1 then $(-1)^{f(0)}$ is either +1 or -1, as a result the first qubit becomes:

$$\begin{aligned}
&= \frac{1}{2}((-1)^{f(0)}(|0\rangle+|1\rangle)+(-1)^{f(1)}(|0\rangle-|1\rangle)) \\
&= \frac{1}{2}((-1)^{f(0)}(|0\rangle+|1\rangle)+(-1)^{f(0)}(|0\rangle-|1\rangle)) \\
&= \frac{1}{2}((-1)^{f(0)}(|0\rangle+|1\rangle+|0\rangle-|1\rangle)) \\
&= (-1)^{f(0)}|0\rangle
\end{aligned}$$

If f is balanced, $f(0)\neq f(1)$,we can conclude that $(-1)^{f(0)}$=$-(-1)^{f(1)}$,then the

first qubit becomes:

$$\begin{aligned} &= \frac{1}{2}((-1)^{f(0)}(|0\rangle + |1\rangle) + (-1)^{f(1)}(|0\rangle - |1\rangle)) \\ &= \frac{1}{2}((-1)^{f(0)}(|0\rangle + |1\rangle) - (-1)^{f(0)}(|0\rangle - |1\rangle)) \\ &= \frac{1}{2}((-1)^{f(0)}(|0\rangle + |1\rangle - |0\rangle + |1\rangle)) \\ &= (-1)^{f(0)}|1\rangle \end{aligned}$$

In short, the state $|\Psi_4\rangle$ is:

$$|\Psi_4\rangle = \begin{cases} |0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right), & if \quad f(0) = f(1) \\ |1\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right), & if \quad f(0) \neq f(1) \end{cases}$$

If we make a measurement of the first qubit it is obvious that we will observe the state $|0\rangle$ if the function is constant, $|1\rangle$ if the function is balanced.

4.1.2. The Deutsch-Jozsa Algorithm

This is a generalized version of Deutsch's algorithm for multiple bit inputs [1, 4, 12]. If we have an input of size n bits then there are two possibilities for the function f. It can be either a constant function giving the same output for each and every input, or a balanced function resulting in 0 for half of the possible inputs and 1 for the other half. If our input consists of n bits than there are 2^n possible inputs ranging from 0 to $2^n - 1$ and the function f outputs 0 for 2^{n-1} of the inputs and 1 for the other 2^{n-1} inputs. We are promised that the third possibility will not occur. The basic assumptions do not change and we still accept that we have the black box U_f working in the same manner, as the one in subsection 4.1.1, $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$.
The Deutsch-Jozsa algorithm is the first algorithm to show the superiority of quantum computation over classical one in the computational complexity sense. If we want to find out whether f is constant or balanced classically then we have to make compu-

tations for $2^{n-1}+1$ times at the worst case. We have to check for half of the inputs' results, if they are the all the same, we have to make one more check, if it is different than the other outputs, we can conclude that the function is balanced, if it is the same we can say that the function f is constant, assuming that there are no other possibilities for the function. However with the quantum Deutsch-Jozsa algorithm we only make one call of the function in the black-box.

1. Initialize the first n qubits to the state $|0\rangle^{\otimes n}$ and last qubit to $|1\rangle$.
2. Apply Hadamard transformation to each of the $n+1$ qubits.
3. Apply the oracle U_f.
4. Apply Hadamard transformation once more to the state in the first n qubits.
5. Measure the value of the first n qubits.
6. If the value is zero then our function is constant, if the measured value is different than zero then the function is balanced.

Figure 4.3. Deutsch-Jozsa Algorithm

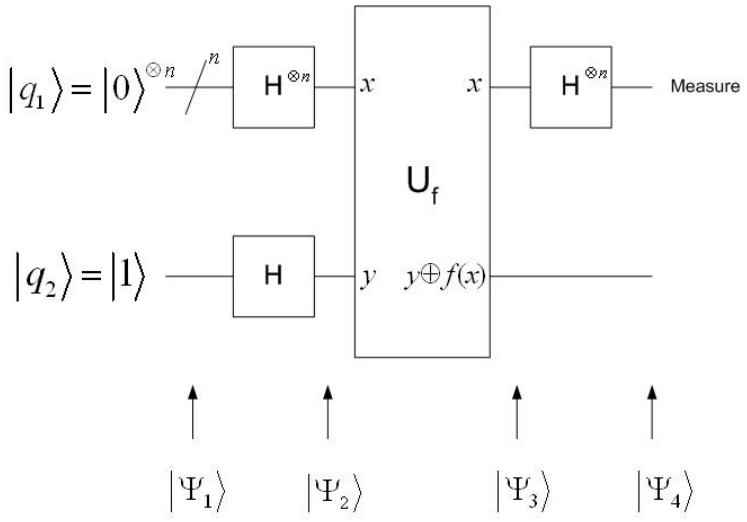

Figure 4.4. Circuit for the Deutsch-Jozsa algorithm

In order to look at how the algorithm works, we can explain it step by step as follows:
1- The initial state for the system is as follows:

$$|q_1\rangle = |0\rangle^{\otimes n} \quad , \quad |q_2\rangle = |1\rangle$$

As a result the total initial state of the system appears as:

$$\begin{aligned} |\Psi_1\rangle &= |q_1\rangle^{\otimes n} \otimes |q_2\rangle \\ &= |0\rangle^{\otimes n} \otimes |1\rangle \end{aligned}$$

2- Application of Hadamard transformation to each qubit results in:

$$\begin{aligned} H|q_1\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \\ H|q_2\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}} \end{aligned}$$

which ends up with the state $|\Psi_2\rangle$ after applying Hadamard transformations to the state $|\Psi_1\rangle$.

$$\begin{aligned} |\Psi_2\rangle &= H^{\otimes n} \otimes H|\Psi_1\rangle \\ &= H^{\otimes n}|q_1\rangle \otimes H|q_2\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \end{aligned}$$

3- Application of the oracle U_f to the system where U_f is accepted to make the transformation $U_f|q_1, q_2\rangle = |q_1, q_2 \oplus f(q_1)\rangle$ without the explicit knowledge of the function f. So the oracle applied form of the system becomes:

$$\begin{aligned} |\Psi_3\rangle &= U_f|\Psi_2\rangle \\ &= U_f \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \frac{1}{\sqrt{2}}((|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle)) \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)}|x\rangle \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} \end{aligned}$$

4- Applying Hadamard transform only to the first n qubits yields:

$$|\Psi_4\rangle = (H^{\otimes n} \otimes I)|\Psi_3\rangle$$

$$
\begin{aligned}
&= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} H^{\otimes n}|x\rangle \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} \\
&= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \sum_{z=0}^{2^n-1} \frac{(-1)^{x\cdot z}|z\rangle}{2^n} \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} \\
&= \sum_{z=0}^{2^n-1} \sum_{x=0}^{2^n-1} \frac{(-1)^{x\cdot z + f(x)}|z\rangle}{2^n} \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}}
\end{aligned}
$$

5- Measure the first n qubits. If we analyze state $|\Psi_4\rangle$ for the cases where f is either constant or balanced considering the amplitude of the $|z\rangle = |0\rangle$ state, it is easy to see that the amplitude is $\sum_{x=0}^{2^n-1} \frac{(-1)^{f(x)}}{2^n}$. If f is constant,then $f(x)$ will be either 0 or 1,the sum of $(-1)^{f(x)}$ values will add up to +1 or -1, if f is balanced then half of the $f(x)$'s will be 0 and the other half will be 1, resulting in half of the $(-1)^{f(x)}$'s being +1 and the other half being -1 respectively, ending up with the cancellation of $(-1)^{f(x)}$'s by one another in the amplitude.

With this result, we can conclude that if f is constant after making the measurement of the first n qubits, with probability 1 we will observe the state $|0\rangle^{\otimes n}$, if the function f is balanced then after the measurement of the first n qubits we will definitely observe a state different than $|0\rangle^{\otimes n}$. Shortly, if we measure all 0s in the n qubits the function f is constant, if we observe a state different than all 0s then the function f is balanced.

4.1.3. CKL for Evenly-Distributed Functions

First the definition of functions which are many-to-one and onto evenly spaced ranges should be done as in [13]. For evenly-distributed functions the domain Z_N is mapped to K evenly spaced range elements in range Z_M where $K \leq M, N$. Since the range is evenly spaced we can say that there exists $\mu = \frac{M}{K}$ separation between the elements. As a result each element in the range can be expressed as $\mu j + t \in Z_M$ where $j \in Z_K$ and t is the possible initial shift as a result $t < \mu$. If we say that our new group has K elements, then N elements in the domain map into these K elements, since our function f is many-to-one and onto, we can say that the domain is mapped to our new range for $\nu = \frac{N}{K}$ times, in other words; ν elements in the domain N maps to the same

element in the range K. In a clearer way, for each element in the range K, there are ν corresponding elements in the domain N. If a function f satisfies these conditions, then f is evenly-distributed and ν-to-1. In this generalization algorithm, the aim is to determine whether f is constant or evenly-distributed with the assumption that there is no other alternatives for f. When K is known, in the worst case $\nu+1$ evaluations are necessary to determine whether f is constant or evenly-distributed. If K is not known then blind trials must be done in the worst case which is exponential as in the original Deutsch-Jozsa problem. In this generalization, in the black-box instead of the classical U_f transformation of Deutsch's algorithm a different transformation which is built up by modifying a known one, which is $U|x\rangle = (-1)^{f(x)}|x\rangle$ is used.The new transformation is $U|x\rangle = \omega_M^{f(x)}|x\rangle$ where $\omega_M = e^{\frac{2\pi i}{M}}$.For simplicity N, M and K are assumed to be powers of two as $N = 2^n$ and $M = 2^m$ and $K = 2^k$.

The procedure is the same as in Deutsch-Jozsa:

1. Initialize the qubits to $|0\rangle^{\otimes n}$.
2. Apply Hadamard to the qubits.
3. Apply the oracle $U|x\rangle = \omega_M^{f(x)}|x\rangle$, defined above.
4. Apply Hadamard again to the qubits.
5. Measure the n qubit state. If the measured state is $|0\rangle$ then f is not evenly-distributed, otherwise f is not constant. If there are only two possibilities for f to be either constant or evenly-distributed then we can conclude that if the measured state is $|0\rangle$ then the function f is constant, otherwise f is evenly-distributed.

Figure 4.5. CKL for Evenly-Distributed functions

The step by step analysis of the algorithm is as follows:

1- Initial state

$$|\Psi_1\rangle \quad = \quad |0\rangle^{\otimes n}$$

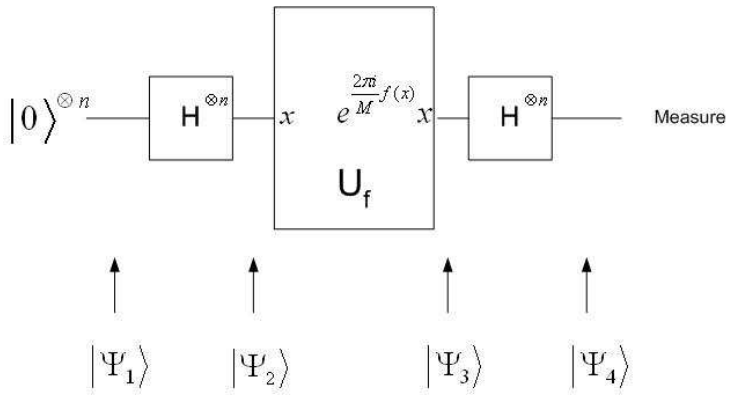

Figure 4.6. Quantum Circuit of CKL for Evenly-Distributed functions

2- Apply Hadamard transformation to all n qubits.

$$\begin{aligned} |\Psi_2\rangle &= H^{\otimes n}|\Psi_1\rangle \\ &= H^{\otimes n}|0\rangle^{\otimes n} \\ &= \frac{1}{\sqrt{N}}\sum_{x=0}^{2^n-1}|x\rangle \end{aligned}$$

3- Application of the new oracle.

$$\begin{aligned} |\Psi_3\rangle &= U|\Psi_2\rangle \\ &= \frac{1}{\sqrt{N}}\sum_{x=0}^{2^n-1}U|x\rangle \\ &= \frac{1}{\sqrt{N}}\sum_{x=0}^{2^n-1}\omega_M^{f(x)}|x\rangle \end{aligned}$$

4- Apply Hadamard to all of the n qubits again.

$$\begin{aligned} |\Psi_4\rangle &= H^{\otimes n}|\Psi_3\rangle \\ &= \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}\omega_M^{f(x)}H^{\otimes n}|x\rangle \\ &= \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}\omega_M^{f(x)}\sum_{z=0}^{N-1}\frac{(-1)^{x\cdot z}|z\rangle}{\sqrt{N}} \\ &= \sum_{z=0}^{N-1}\left\{\frac{1}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot z}\omega_M^{f(x)}\right\}|z\rangle \end{aligned}$$

As a result when the amplitude of any state of $|z\rangle$ is analyzed, it is clear that:

$$S_z = \frac{1}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot z}\omega_M^{f(x)}$$

If f is constant then $f(0) = f(x)$
for $z = 0$

$$\begin{aligned} S_0 &= \frac{1}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot 0}\omega_M^{f(0)} \\ &= \omega_M^{f(0)}\frac{1}{N}\sum_{x=0}^{N-1}1 \\ &= \omega_M^{f(0)}\frac{N}{N} \\ &= \omega_M^{f(0)} \end{aligned}$$

for $z \neq 0$

$$\begin{aligned} S_z &= \frac{1}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot z}\omega_M^{f(0)} \\ &= \frac{\omega_M^{f(0)}}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot z} \\ &= \frac{\omega_M^{f(0)}}{N}0 \\ &= 0 \end{aligned}$$

$\sum_{x=0}^{N-1}(-1)^{x\cdot z}$ results in zero, since half of the $(-1)^{x\cdot z}$ will result in -1 and the other half will result in 1 when $z \neq 0$ and x is variant in the range $[0, 2^n - 1]$.

As a result when f is constant:

$$\begin{aligned} S_z &= \omega_M^{f(0)}\frac{1}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot z} \\ &= \begin{cases} \omega_M^{f(0)}, & z = 0; \\ 0, & z \neq 0. \end{cases} \end{aligned}$$

When f is evenly-distributed, meaning that $N = \nu K$ and $f(x) = \mu j + t$:
for $z = 0$

$$\begin{aligned} S_0 &= \frac{1}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot 0}\omega_M^{f(x)} \\ &= \frac{1}{\nu K}\nu\sum_{j=0}^{K-1}\omega_M^{\mu j+t} \\ &= \frac{1}{K}\omega_M^t\sum_{j=0}^{K-1}\omega_K^j \\ &= \frac{1}{K}\omega_M^t 0 \\ &= 0. \end{aligned}$$

As a result, we have three possibilities as follows:

$$S_z = \begin{cases} \omega_M^{f(0)}, & f \text{ is constant and } z = 0; \\ 0 \quad , & f \text{ is constant and } z \neq 0; \\ 0 \quad , & f \text{ is evenly-distributed and } z = 0. \end{cases}$$

To sum up, if we measure the state $|z\rangle = |0\rangle$ then f is not evenly-distributed. If we measure $|z\rangle \neq |0\rangle$ then f is not constant.

4.1.4. CKL for Evenly-Balanced Functions

In the original Deutsch-Jozsa algorithm the function f is evenly-balanced if half of the outputs are 0 and half of them are 1. The first difference in CKL [13] is that the function f may return m-bit outputs such that half of them have the parity 0 and the other half have the parity 1. Therefore, in this algorithm we have an n qubit control register and an m qubit auxiliary register. As a result, instead of the direct output of the function, we check for the parity of the m-bit second register as stated that half of the outputs of the function have the parity 0 and the other half have the parity 1. We prepare both registers and indicate $N = 2^n$ and $M = 2^m$. We again have the oracle and we use the one of the original Deutsch's algorithm which works as

$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. In this new algorithm, we are promised to deal with a function f which is either constant or evenly-balanced assuming that no other possibilities exist for f. The algorithm below determines whether the function f is constant or evenly-balanced with two calls of the function in the oracle calls. The algorithm is as follows:

1. Initialize the n qubit control register to $|0\rangle^{\otimes n}$
2. Apply Hadamard transformation to the n-qubit control register.
3. Apply the oracle U_f as in the original Deutsch-Jozsa algorithm.
4. Apply Pauli-spin matrix corresponding to phase-flip operator $\sigma_z^{\otimes m}$ to the m-qubit auxiliary register.
5. Apply the oracle U_f to the control register again.
6. Apply phase-flip operator to the auxiliary register once more.
7. Apply Hadamard transformation to the control register once more.
8. Measure the n-qubit control register. If the output state is $|0\rangle$ then function f is constant, if the output state is not $|0\rangle$ then the function f is balanced.

Figure 4.7. CKL for Evenly-Balanced functions

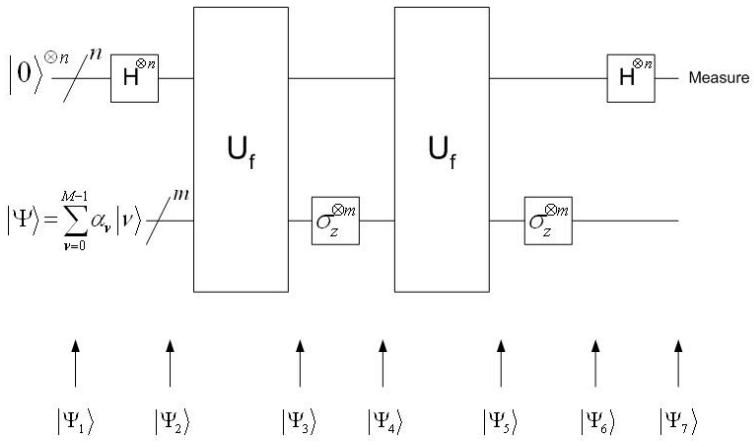

Figure 4.8. Quantum Circuit of CKL for Evenly-Balanced functions

In order to analyze the steps of the algorithm we start with the preparation of the registers. We assume that the auxiliary register is the combination of superpositions of

all possible m bit vectors with arbitrary amplitudes such that their probabilities add up to 1 and can be denoted as follows:

$$\begin{aligned} |\psi\rangle &= \sum_{\nu=0}^{M-1} \alpha_\nu |\nu\rangle \\ &= \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \end{aligned}$$

Step by step analysis:

1- Initialize the n-qubit control register to $|0\rangle^{\otimes n}$

$$|\psi_1\rangle = |0\rangle^{\otimes n} \otimes |\psi\rangle$$

2- Apply Hadamard transformation to the n-qubit control register.

$$\begin{aligned} |\psi_2\rangle &= \left(H^{\otimes n} \otimes I^{\otimes m}\right) |\psi_1\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \otimes |\psi\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \otimes \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \end{aligned}$$

3- Apply the oracle U_f as in the original Deutsch-Jozsa algorithm.

$$\begin{aligned} |\psi_3\rangle &= U_f |\psi_2\rangle \\ &= U_f \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \otimes \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \otimes \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j \oplus f(x)_j\rangle \end{aligned}$$

where $f(x) = \sum_{j=0}^{m-1} f(x)_j 2^j$.

4- Apply the Pauli-spin matrix corresponding to the phase-flip operator $\sigma_z^{\otimes m}$ to the m-qubit auxiliary register.

We already know from section 2.1 that for one qubit operation the Pauli spin matrix

phase-flip operator has the effect:

$$\begin{aligned}\sigma_z|0\rangle &= |0\rangle \\ \sigma_z|1\rangle &= -|1\rangle\end{aligned}$$

When $|\nu_j \oplus f(x)_j\rangle$ is analyzed bit by bit for total of m bits, it is clear that the bits in state $|1\rangle$ will bring -1 as the coefficient and the bits in state $|0\rangle$ will bring 1. Therefore, the total effect of the phase-flip Pauli spin operator to the state $|\nu_j \oplus f(x)_j\rangle$ is $(-1)^{\sum_{j=0}^{m-1}(v_j+f(x)_j)}$, since as a power of (-1), $\sum_{j=0}^{m-1}(v_j + f(x)_j)$ and $\sum_{j=0}^{m-1}(v_j \oplus f(x)_j)$ have no difference.

$$\begin{aligned}|\psi_4\rangle &= \left(I^{\otimes n} \otimes \sigma_z^{\otimes m}\right)|\psi_3\rangle \\ &= \left(I^{\otimes n} \otimes \sigma_z^{\otimes m}\right)\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle \otimes \sum_{\nu=0}^{M-1}\alpha_\nu \bigotimes_{j=0}^{m-1}|\nu_j \oplus f(x)_j\rangle \\ &= \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle \otimes \sum_{\nu=0}^{M-1}(-1)^{\sum_{j=0}^{m-1}(v_j+f(x)_j)}\alpha_\nu \bigotimes_{j=0}^{m-1}|\nu_j \oplus f(x)_j\rangle \\ &= \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}(-1)^{\sum_{j=0}^{m-1}f(x)_j}|x\rangle \otimes \sum_{\nu=0}^{M-1}(-1)^{\sum_{j=0}^{m-1}v_j}\alpha_\nu \bigotimes_{j=0}^{m-1}|\nu_j \oplus f(x)_j\rangle\end{aligned}$$

5- Apply the oracle $U_f^{\otimes n}$ to the control register again.

$$\begin{aligned}|\psi_5\rangle &= U_f|\psi_4\rangle \\ &= U_f\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}(-1)^{\sum_{j=0}^{m-1}f(x)_j}|x\rangle \otimes \sum_{\nu=0}^{M-1}(-1)^{\sum_{j=0}^{m-1}v_j}\alpha_\nu \bigotimes_{j=0}^{m-1}|\nu_j \oplus f(x)_j\rangle \\ &= \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}(-1)^{\sum_{j=0}^{m-1}f(x)_j}|x\rangle \otimes \sum_{\nu=0}^{M-1}(-1)^{\sum_{j=0}^{m-1}v_j}\alpha_\nu \bigotimes_{j=0}^{m-1}|\nu_j \oplus f(x)_j \oplus f(x)_j\rangle \\ &= \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}(-1)^{\sum_{j=0}^{m-1}f(x)_j}|x\rangle \otimes \sum_{\nu=0}^{M-1}(-1)^{\sum_{j=0}^{m-1}v_j}\alpha_\nu \bigotimes_{j=0}^{m-1}|\nu_j\rangle\end{aligned}$$

6-Apply phase-flip operator to the auxiliary register once more.

$$|\psi_6\rangle = \left(I^{\otimes n} \otimes \sigma_z^{\otimes m}\right)|\psi_5\rangle$$

$$
\begin{aligned}
&= \left(I^{\otimes n} \otimes \sigma_z^{\otimes m}\right) \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes \sum_{\nu=0}^{M-1} (-1)^{\sum_{j=0}^{m-1} v_j} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes \sum_{\nu=0}^{M-1} (-1)^{2\sum_{j=0}^{m-1} v_j} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes |\psi\rangle
\end{aligned}
$$

7- Apply Hadamard transformation to the control register once more.

$$
\begin{aligned}
|\psi_7\rangle &= \left(H^{\otimes n} \otimes I^{\otimes m}\right) |\psi_6\rangle \\
&= \left(H^{\otimes n} \otimes I^{\otimes m}\right) \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes |\psi\rangle \\
&= \sum_{y=0}^{N-1} \left\{ \frac{1}{N} \sum_{x=0}^{N-1} (-1)^{x\cdot y} (-1)^{\sum_{j=0}^{m-1} f(x)_j} \right\} |y\rangle \otimes |\psi\rangle
\end{aligned}
$$

If we analyze the amplitude S_y of the output state $|y\rangle$ for both f is constant and f is balanced we get:
f is constant, meaning $f(x) = f(0)$ all of the $f(x)$'s have the same parity 0 or 1 for $y = 0$

$$
\begin{aligned}
S_y &= \frac{1}{N} \sum_{x=0}^{N-1} (-1)^{x\cdot y} (-1)^{\sum_{j=0}^{m-1} f(0)_j} \\
S_0 &= \frac{(-1)^{\sum_{j=0}^{m-1} f(0)_j}}{N} \sum_{x=0}^{N-1} 1 \\
&= (-1)^{\sum_{j=0}^{m-1} f(0)_j}
\end{aligned}
$$

Depending on the common parity of all the f's (either 0 or 1) $(-1)^{\sum_{j=0}^{m-1} f(0)_j}$ is either -1 or 1. The probability is $|S_0|^2 = 1$.

for $y \neq 0$

$$\begin{aligned} S_y &= \frac{1}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot y}(-1)^{\sum_{j=0}^{m-1} f(0)_j} \\ &= \frac{(-1)^{\sum_{j=0}^{m-1} f(0)_j}}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot y} \\ &= \frac{(-1)^{\sum_{j=0}^{m-1} f(0)_j}}{N}0 \\ &= 0 \end{aligned}$$

For f is constant

$$S_y = \begin{cases} 0, & if\ y \neq 0; \\ 1, & if\ y = 0. \end{cases} \tag{4.1}$$

If we analyze the same probability for the state $|y\rangle$ in case f is balanced then we have:

for $y = 0$

$$\begin{aligned} S_y &= \frac{1}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot y}(-1)^{\sum_{j=0}^{m-1} f(x)_j} \\ S_0 &= \frac{1}{N}\sum_{x=0}^{N-1}(-1)^{\sum_{j=0}^{m-1} f(x)_j} \\ &= \frac{1}{N}\sum_{x=0}^{N-1}0 \\ &= 0 \end{aligned}$$

As a result, what we have in our hands at the end of the algorithm is:

$$S_y = \begin{cases} 1, & f \text{ is constant and } y = 0; \\ 0, & f \text{ is constant and } y \neq 0; \\ 0, & f \text{ is balanced and } y = 0. \end{cases}$$

In short, it is easy to say that if we measure $|0\rangle$ as the output state then the function f is constant, if we measure a state other than $|0\rangle$ state, then the function f is evenly-balanced.

4.2. Ballhysa's Generalization of the Deutsch-Jozsa algorithm

As in the Deutsch-Jozsa problem, we have a function f such that $f : \{0,1\}^n \rightarrow \{0,1\}$ and we want to find out whether f is constant or balanced in this algorithm [14, 15]. The difference from the original Deutsch-Jozsa problem is that the domain consists of two sets say S and S' complementing each other in the n-bit domain. However we assume that our function f has its typical properties in the set S such that the number of elements of S is greater than 2^{n-1}. If we say that the cardinality of the set S is q and these numbers are in the interval $U_n = [0, 2^n - 1]$, we can make clear definition of the assumptions for our new domain:

1. q is even and greater than 2^{n-1}
2. Function f is balanced on the set $S' = U_n - S$
3. Function f is either balanced or constant on the set S.

If we say that $S = U_n$ then we go back to the original Deutsch-Jozsa algorithm, and it becomes a special case of the above conditions. With the conditions above we are still to find whether f is constant or balanced on the set S. However this time instead of using Hadamard gates to create equal superpositions of all n bit numbers, we will use Equiprobable Superposition Generator, ESG, for q numbers which is not necessarily a power of 2. As a result, the circuit is almost the same as the original Deutsch-Jozsa circuit, with the difference that instead of the first Hadamard gate to n qubits, we have the Equiprobable Superposition Generator applied to the n qubits. At the first stage we assume that the Equiprobable Superposition Generator works perfectly such that for each element in the domain set S whose cardinality is q,it creates perfectly equal amplitudes of $\frac{1}{\sqrt{q}}$. The algorithm is as follows:

1. Initialize the first n qubits to the state $|0\rangle^{\otimes n}$ and second qubit to $|1\rangle$.
2. Apply Equiprobable Superposition Generator to first n qubits and Hadamard gate to the last qubit.
3. Apply the oracle U_f which is previously defined in Deutsch-Jozsa algorithm.
4. Apply Hadamard transformation to the state in the first n qubits.
5. Measure the value of the first n qubits. If the measured value is different than zero, then the function f is claimed to be balanced. If the measured value is zero then the function is constant.

Figure 4.9. Quantum algorithm for the Generalized Deutsch-Jozsa Algorithm by Ballhysa

The step by step explanation of the algorithm with the error probabilities is follows:

1- Initialize the first n qubits to the state $|0\rangle^{\otimes n}$ and second qubit to $|1\rangle$.

$$|\psi_1\rangle = |0\rangle^{\otimes n} \otimes |1\rangle$$

2- Apply Equiprobable Superposition Generator to each n qubits and Hadamard gate to the last qubit.

$$\begin{aligned} |\psi_2\rangle &= (ESG \otimes H)|\psi_1\rangle \\ |\psi_2\rangle &= (ESG \otimes H)(|0\rangle^{\otimes n} \otimes |1\rangle) \\ &= \left[\frac{1}{\sqrt{q}} \sum_{x \in S} |x\rangle\right] \otimes \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] \end{aligned}$$

3- Apply the oracle U_f which is previously defined in Deutsch-Jozsa algorithm.

$$\begin{aligned} |\psi_3\rangle &= U_f|\psi_2\rangle \\ &= U_f \left(\left[\frac{1}{\sqrt{q}} \sum_{x \in S} |x\rangle\right] \otimes \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]\right) \\ &= \left[\frac{1}{\sqrt{q}} \sum_{x \in S} (-1)^{f(x)}|x\rangle\right] \otimes \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] \end{aligned}$$

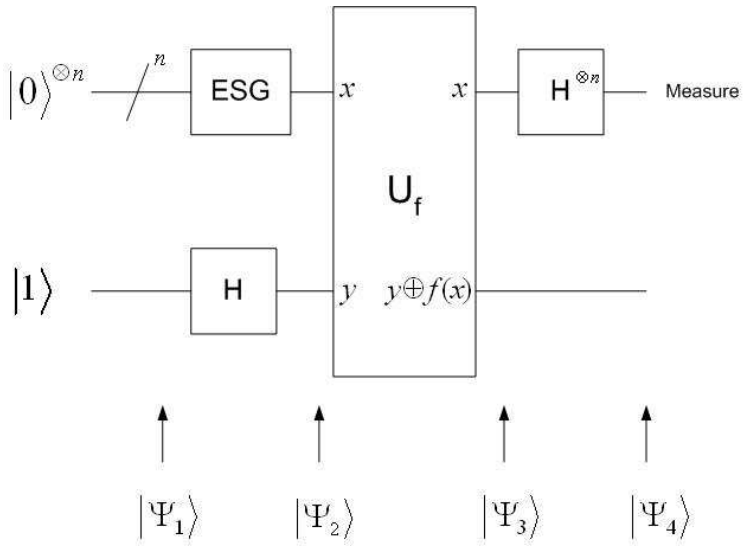

Figure 4.10. Quantum Circuit of the Generalized Deutsch-Jozsa algorithm by Ballhysa

4- Apply Hadamard transformation to the state in the first n qubits.

$$\begin{aligned} |\psi_4\rangle &= H^{\otimes n} \otimes I |\psi_3\rangle \\ &= H^{\otimes n} \otimes I \left[\frac{1}{\sqrt{q}} \sum_{x \in S} (-1)^{f(x)} |x\rangle\right] \otimes \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] \\ &= \frac{1}{\sqrt{2^n}} \sum_{z=0}^{2^n-1} \left[\frac{1}{\sqrt{q}} \sum_{x \in S} (-1)^{x \cdot z} (-1)^{f(x)} |x\rangle\right] \otimes \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] \end{aligned}$$

5- Measure the value of the first n qubits.

If we look for the amplitude, S_z, for any measurable state $|z\rangle$ then we will see that:

$$S_z = \frac{1}{\sqrt{2^n}} \left[\frac{1}{\sqrt{q}} \sum_{x \in S} (-1)^{x \cdot z} (-1)^{f(x)}\right]$$

If we look for the state $|z\rangle = |0\rangle$ then,

$$S_0 = \frac{1}{\sqrt{2^n}} \left[\frac{1}{\sqrt{q}} \sum_{x \in S} (-1)^{f(x)}\right] \tag{4.2}$$

If the function f is balanced then $\sum_{x \in S} (-1)^{f(x)}$ yields 0, as a result the probability to observe $|0\rangle$ state becomes 0.

If the function f is constant,$f(x) = f(0)$, then the amplitude becomes:

$$\begin{aligned} S_0 &= \frac{1}{\sqrt{2^n}} \left[\frac{1}{\sqrt{q}} \sum_{x \in S} (-1)^{f(0)} \right] \\ &= \frac{(-1)^{f(0)}}{\sqrt{2^n}} \left[\frac{1}{\sqrt{q}} \sum_{x \in S} 1 \right] \\ &= \frac{(-1)^{f(0)}}{\sqrt{2^n}} \left[\frac{1}{\sqrt{q}} \right] q \\ &= (-1)^{f(0)} \left[\sqrt{\frac{q}{2^n}} \right] \end{aligned}$$

As a result, the probability, P_0, which is the square of the amplitude, S_0, is:

$$\begin{aligned} P_0 &= |S_0|^2 \\ &= \left((-1)^{f(0)} \left[\sqrt{\frac{q}{2^n}} \right] \right)^2 \\ &= \frac{q}{2^n} \end{aligned}$$

If we could obtain perfect equiprobable superpositions in step 2 of the algorithm, then with probability $\frac{q}{2^n}$ we could observe $|0\rangle$ state when the function f is constant. Since q is greater than 2^{n-1} then this probability is greater than $\frac{1}{2}$. This algorithm is not an exact algorithm as the original Deutsch-Jozsa algorithm, however it has the 'one-sided error' property, such that when the function f is balanced, it is classified correctly with certainty. As mentioned in the beginning of this section we assumed that ESG, creates perfectly equal superpositions of the set S. However in practice this is not possible. There are two algorithms, implemented for ESG. One of them is due to Kitaev *et al.* and the other is Algorithm G and it is based on Grover's quantum search algorithm. They will not be discussed here in detail, details can be found in [14].

4.2.1. KSV and Ballhysa's Algorithm with Algorithm KSV

KSV(Kitaev, Shen and Vyalyi Algorithm) [14] works for the set Q which consists of the superpositions of states $Q = \{0, 1, ..., q-1\}$ and creates a superposition of each

state as:

$$|\psi\rangle \quad = \quad \sum_{i=0}^{q-1} \delta_i |i\rangle$$

Each δ_i is not equal to $1/\sqrt{q}$, however an approximate value of $1/\sqrt{q}$.

As understood, the set Q is an ordered set but our set S includes numbers with no order, as a result after processing KSV, a permutation operator P_s is applied and each element i of the set Q is mapped to an element s_i of set S with their amplitudes. At the end, an approximate equiprobable superposition of set S is:

$$|\psi'\rangle \quad = \quad \sum_{x_i \in S} \delta_i |x_i\rangle \tag{4.3}$$

For Ballhysa's algorithm analysis, the amplitude of the state $|0\rangle$ is as in equation 4.2:

$$S_0 \quad = \quad \frac{1}{\sqrt{2^n}} \left[\frac{1}{\sqrt{q}} \sum_{x \in S} (-1)^{f(x)} \right]$$

With KSV the amplitude S_0 is:

$$S_0 \quad = \quad \frac{1}{\sqrt{2^n}} \left[\sum_{x_i \in S} (-1)^{f(x_i)} \delta_i \right]$$

For a balanced function f, we expect the sum to vanish each other with perfect equiprobable superpositions, however with KSV although the amplitudes are very close to $1/\sqrt{q}$, they are different and they will not add up to 0. There will be a small number which is close to 0. Therefore with KSV, one-sided error property of Ballhysa's original algorithm will be lost.

4.2.2. Algorithm G and Ballhysa's Algorithm with Algorithm G

For G algorithm [14] which is based on Grover's search an oracle is defined for the Grover's search process. After the application of Grover's search algorithm a predefined permutation operator is applied to construct a mapping from the ordered set Q to set S as in KSV. The resulting solution is different from KSV as:

$$|\psi\rangle \quad = \quad a \sum_{x \in S} |x\rangle + b \sum_{x \notin S} |x\rangle$$

where a is an approximate value of $1/\sqrt{q}$, and b is an approximate value of 0.

The amplitude of state $|0\rangle$ as in equation 4.2 in a perfectly equiprobable generated states is:

$$S_0 \quad = \quad \frac{1}{\sqrt{2^n}} \left[\frac{1}{\sqrt{q}} \sum_{x \in S} (-1)^{f(x)} \right]$$

In the original algorithm we expect this amplitude will vanish with cancellations when the function is balanced.

With Algorithm G the amplitude will be:

$$S_0 = \frac{1}{\sqrt{2^n}} \left[a \sum_{x \in S} (-1)^{f(x)} + b \sum_{x \notin S} (-1)^{f(x)} \right] \tag{4.4}$$

We made the assumption that the function f is balanced on S' which is set of n-bit numbers which are not in the set S. Therefore $b \sum_{x \notin S} (-1)^{f(x)}$ part of the equation 4.4 will vanish because of the cancellations just like $a \sum_{x \in S} (-1)^{f(x)}$ part. As a result, the probability to observe the state $|0\rangle$ when the function f is balanced will remain 0. Therefore, the one-sided property of Ballhysa's algorithm survives with Algorithm G.

When the function is constant, the probability of observing the state $|0\rangle$ will be $qp/2^n$ where p is the success probability of Grover's algorithm. This probability can be increased by adding extra bits to the Grover's algorithm. However adding extra bits will slow down the running time of the algorithm. Details can be found in [14].

4.3. Simon's Algorithm

Simon's problem [5] is a generalized version of Deutsch's problem. We suppose that we have a function $f : \{0,1\}^n \rightarrow \{0,1\}^m$,with $m \geq n$, then the xor-mask invariance of f is:

- s, if there exists a non–trivial string s of length n such that $\forall\ f(x) = f(x \oplus s)$.
- 0^n, if f is 1-to-1.
- undefined otherwise.

The problem is to determine whether function f is 1-to-1. If f is not 1-to-1, then the aim is to find s. We are promised that the remaining possibility does not occur. If there exists the string s, it is clear that our function is 2-to-1, since there will be two strings in the domain separated by the string s mapping to the same output in the range.

The algorithm is given in Figure 4.11:

1. Initialize totally $n+m$ qubits, n qubits for the first register and m qubits for the second one, to the state $|0\rangle^{\otimes n+m}$
2. Apply Hadamard transformation to the first register.
3. Apply oracle U_f defined similarly to the one in Deutsch's algorithm rewritten for multiple qubit target register as $U_f|x\rangle^{\otimes n}|y\rangle^{\otimes m} = |x\rangle|y \oplus f(x)\rangle$.
4. Apply Hadamard transformation to the first register again.
5. Measure the first register and put the measured string to a row of the matrix A of step 6.
6. Repeat the first five steps $n-1$ times to constitute an $n \times n$ non-singular matrix A with output strings each measured in the first n-qubit register as row vectors of the matrix.
7. Solve the matrix problem $A_{(n-1)\times n} s_{n\times 1} = 0$ by using Gaussian elimination.
8. Check for the property of f by evaluating $f(0)$ and $f(s)$.

Figure 4.11. Simon's Algorithm

The analysis of the algorithm step by step is as follows:

1- Initialize totally $n+m$ qubits, n qubits for the first register and m qubits for the second one, to the state $|0\rangle^{\otimes(n+m)}$

$$|\psi_1\rangle = |0\rangle^{\otimes(n+m)}$$

2- Apply Hadamard transformation to the first register.

$$\begin{aligned} |\psi_2\rangle &= \left(H^{\otimes n} \otimes I^{\otimes m}\right)|\psi_1\rangle \\ &= \left(H^{\otimes n} \otimes I^{\otimes m}\right)|0\rangle^{\otimes(n+m)} \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |0\rangle^{\otimes m} \end{aligned}$$

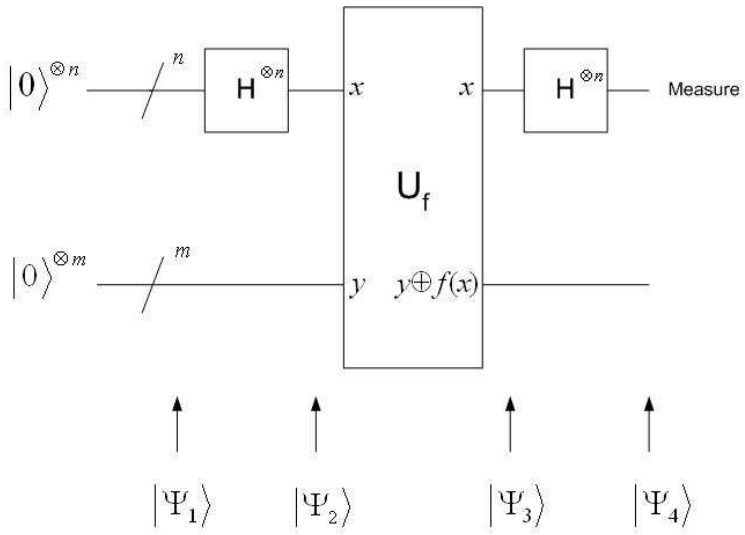

Figure 4.12. Quantum Circuit for the first four steps of Simon's problem

3- Apply the oracle U_f defined as $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$.

$$\begin{aligned} |\psi_3\rangle &= U_f|\psi_2\rangle \\ &= U_f \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |0\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |0 \oplus f(x)\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |f(x)\rangle \end{aligned}$$

4- Apply Hadamard transformation to the first register again.

$$\begin{aligned} |\psi_4\rangle &= \left(H^{\otimes n} \otimes I^{\otimes m}\right) |\psi_3\rangle \\ &= \left(H^{\otimes n} \otimes I^{\otimes m}\right) \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |f(x)\rangle \\ &= \frac{1}{2^n} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle \otimes |f(x)\rangle \end{aligned}$$

Now if we suppose that the string s exists and $\forall x \neq x'$

$$f(x) = f(x') \Leftrightarrow x' = x \oplus s$$

Then if we compute the amplitude of the state $|y\rangle \otimes |f(x)\rangle$:

$$S_y = \frac{1}{2^n}\left((-1)^{x\cdot y} + (-1)^{(x\oplus s)\cdot y}\right)$$

We know that:

$$\begin{aligned}
(x \oplus s)\cdot y &= (x_0 \oplus s_0)y_0 + (x_1 \oplus s_1)y_1 + ... + (x_{n-1} \oplus s_{n-1})y_{n-1} \qquad mod\ 2 \\
&= x_0y_0 + s_0y_0 + x_1y_1 + s_1y_1 + ... + x_{n-1}y_{n-1} + s_{n-1}y_{n-1} \quad mod\ 2 \\
&= x_0y_0 + x_1y_1 + ... + x_{n-1}y_{n-1} + s_0y_0 + s_1y_1 + ... + s_{n-1}y_{n-1} \quad mod\ 2 \\
&= x\cdot y + s\cdot y \qquad mod\ 2
\end{aligned}$$

If $s \cdot y \neq 0$ then the amplitude

$$\begin{aligned}
S_y &= \frac{1}{2^n}\left((-1)^{x\cdot y} + (-1)^{(x\oplus s)\cdot y}\right) \\
&= \frac{1}{2^n}\left((-1)^{x\cdot y} + (-1)^{x\cdot y + s\cdot y}\right) \\
&= \frac{1}{2^n}\left((-1)^{x\cdot y} + (-1)^{x\cdot y + 1}\right) \\
&= 0
\end{aligned}$$

As a result, what we will measure in the first register at the end will be with certainty an n bit string t such that $t \cdot s = 0$.

5- Measure the first register and put the measured string to a row of the matrix A of step 6. Each measurement will constitute a row in the $A_{(n-1)\times n}$ matrix with the output string t which has the property $t \cdot s = 0$. $n-1$ such strings will give us the opportunity to apply Gaussian elimination to find the string s.

6- Repeat the first five procedures for $n-1$ times to get an independent set from the output of the first register.

When we repeat the algorithm for $n-1$ times with at least $\frac{1}{4}$ probability an independent set of t strings can be gathered.

Lemma If s is a non-zero binary vector y of length n, then $n-1$ randomly chosen

binary vectors of length n such that $s \cdot y \equiv 0 \mod 2$ are linearly independent with probability at least 1/4.

Proof:

Suppose that our $n-1$ independent vectors are $y_1, y_2, ..., y_{n-1}$. For any $i-1$ vectors, namely $y_1, y_2, ..., y_{i-1}$,where $1 < i < n$, at most 2^{i-1} vectors are linear combinations of them. Since the maximum number of n bit vectors y such that $y \cdot s = 0$ is 2^{n-1}, the minimum number of vectors that are not the combinations of $i-1$ vectors becomes $2^{n-1} - 2^{i-1}$. As a result, the probability that the i^{th} vector y_i is independent from the first $i-1$ vectors is at least:

$$\frac{2^{n-1} - 2^{i-1}}{2^{n-1}} = 1 - \frac{1}{2^{n-i}}$$

When we apply the above formula for all vectors from y_1 to y_{n-1} then the probability that we have $n-1$ independent vectors, P_{n-1}, becomes:

$$\left(1 - \frac{1}{2^{n-1}}\right)\left(1 - \frac{1}{2^{n-2}}\right)...\left(1 - \frac{1}{2}\right) \tag{4.5}$$

We already know the rule:
For the numbers $a_1, a_2, ..., a_k$ such that $\forall i$,where $1 \leq i \leq k$, $0 < a_i$ and $\sum_{i=1}^{k} a_i < 1$

$$(1 - a_1)(1 - a_2)...(1 - a_k) \geq 1 - (a_1 + a_2 + ... + a_k)$$

Another rule we know is:

$$\frac{1}{2^2} + \frac{1}{2^3} + ... + \frac{1}{2^{n-1}} < \frac{1}{2}$$

When we apply the above two rules to the equation 4.5:

$$\begin{aligned} P_{n-1} &= \left(1-\frac{1}{2^{n-1}}\right)\left(1-\frac{1}{2^{n-2}}\right)...\left(1-\frac{1}{2}\right) \\ &\geq \left(1-\left(\frac{1}{2^2}+\frac{1}{2^3}+...+\frac{1}{2^{n-1}}\right)\right)\left(1-\frac{1}{2}\right) \\ &> \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) \\ &= \frac{1}{4} \end{aligned}$$

As a result, we can say that repeating the first four steps for $n-1$ times will succeed in finding all independent vectors at least with the probability $\frac{1}{4}$.

6- Solve the matrix problem $A_{n-1\times n}s_{n\times 1} = 0$ by using Gaussian elimination. The method is explained in Appendix A.

7- Check for the property of f by evaluating $f(0)$ and $f(s)$.

If the output of the second register differs for $f(x)$ and $f(x\oplus s)$, then the function is 1-to-1, if the outputs are the same then the function is 2-to-1. An exact polynomial-time algorithm for Simon's problem has also been developed and can be found in [16].

4.4. Factorization

4.4.1. Quantum Fourier Transform

One of the applications used in mathematics is the Fourier Transformation [1, 11, 12]. The Fourier transformation is used to transform functions of the time domain to the frequency domain. In other words, it maps functions with period r to the functions with non-zero values only at the multiples of the frequency $\frac{1}{r}$. The Discrete Fourier Transform is applied to an input vector $x = [x_0, x_1, ..., x_{N-1}]$ of length N and results in the output vector $y = [y_0, y_1, ..., y_{N-1}]$ of length N where the components of the output vector y are defined as:

$$y_j \equiv \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_k \; e^{\frac{2\pi\imath jk}{N}} \tag{4.6}$$

The Quantum Fourier Transform, QFT, has as basis the Discrete Fourier Transform. The only difference is that instead of the vector x the input will be a basis vectors from $|0\rangle$ to $|N-1\rangle$, the vectors in the group Z_N, where N can be accepted to be a power of two, $N = 2^n$, without loss of generality. As a result, QFT is the unitary transformation with the following formula:

$$QFT|k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{2\pi\imath jk}{N}} |j\rangle$$

where $0 \leq k < N$. The matrix representation of the Quantum Fourier Transformation is as follows:

$$QFT_N = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{(N-1)} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ 1 & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)^2} \end{bmatrix}$$

where $\omega = e^{\frac{2\pi\imath}{N}}$ is the N^{th} root of unity. As a result, the effect of the Quantum Fourier transform on an arbitrary state will be:

$$QFT\left(\sum_{k=0}^{N-1} x_k|k\rangle\right) = \sum_{j=0}^{N-1} y_j|j\rangle$$

where y_j and x_k are the coefficients of the Discrete Fourier Transformation equation 4.6.

The gates R_k in Figure 4.13 denote the unitary transformation:

$$R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi\imath}{2^k}} \end{bmatrix}$$

which corresponds to *conditional phase shift* by the angle $e^{\frac{2\pi\imath}{2^k}}$. When we search for the number of gates the circuit uses, we see that for n qubits starting from the first one to

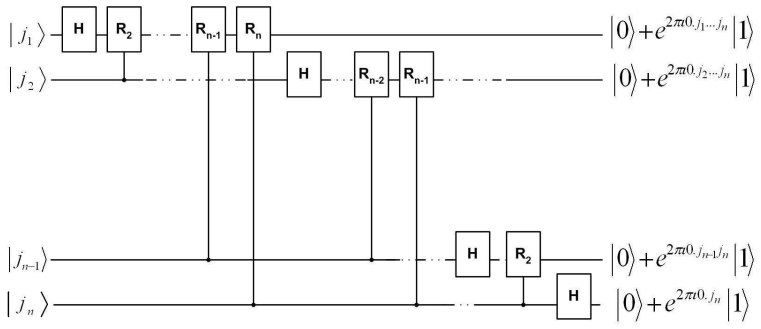

Figure 4.13. Quantum Circuit for Quantum Fourier Transformation

the last,we use $n-k$ R gates and a Hadamard gate. As a result, total number of gates used becomes:

$$\begin{aligned}\sum_{k=1}^{n}(n-k+1) &= n+(n-1)+(n-2)+...+1 \\ &= \frac{n(n+1)}{2}\end{aligned}$$

In addition, in order to have the correct order of the qubits we need at most $\frac{n}{2}$ swaps, each of which can be handled using three controlled-NOT gates. Therefore, we need $O(n^2)$ gates to perform the Quantum Fourier Transformation. However, the best classical algorithm to compute Discrete Fourier Transformation on 2^n elements is called *Fast Fourier Transformation* and computes using $\Theta(n2^n)$ gates. When compared, it is obvious that classical Fast Fourier Transformation needs more operations to handle Fourier Transformation.

4.4.2. Phase Estimation

The Phase Estimation algorithm is a very essential algorithm especially because of the basis it prepares for Shor's Factorization Algorithm [1, 6]. Another important issue about the Phase Estimation Algorithm is that it lies on the basis of Quantum Fourier Transformation. We use Phase Estimation, where we have a unitary operator U with an eigenvector $|u\rangle$ and eigenvalue $e^{2\pi\imath\varphi}$ with an unknown φ. Here we assume that we have a black box to perform the controlled-U^j operation, for possible non-negative

integers j. Controlled-U^j as understood from its name, applies the unitary operator U^j according to the state of j of the control register, where $1 \leq j \leq n$. The algorithm for Phase Estimation is in Figure 4.14.

1. Initial state is the preparation of t qubits $|0\rangle^{\otimes t}$ and the eigenvector $|u\rangle$ which is already assumed to be known, in two registers.
2. Apply Hadamard to the first register of t qubits of $|0\rangle^{\otimes t}$ state.
3. Apply the black box controlled-U^j.
4. Apply the inverse Fourier transform.
5. Measure the first register and find an approximation to the phase.

Figure 4.14. Phase Estimation Algorithm

When we make the step by step analysis of the algorithm.
1- Initial state is the preparation of t qubits $|0\rangle^{\otimes t}$ and the eigenvector $|u\rangle$ which is already assumed to be known.

$$|\psi_1\rangle = |0\rangle^{\otimes t} \otimes |u\rangle$$

2- Apply Hadamard to the first register of t qubits of $|0\rangle^{\otimes t}$ state.

$$\begin{aligned} |\psi_2\rangle &= \left(H^{\otimes t}|0\rangle\right)|u\rangle \\ &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|u\rangle \end{aligned}$$

3- Apply the black box controlled-U^j.

$$\begin{aligned} |\psi_3\rangle &= controlled - U^j \left(\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|u\rangle\right) \\ &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle U^j|u\rangle \\ &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle e^{2\pi\imath j\varphi}|u\rangle \end{aligned}$$

$$= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi\imath j\varphi}|j\rangle|u\rangle$$

4- Apply the inverse Fourier transform.

$$\begin{aligned} |\psi_4\rangle &= QFT^\dagger|\psi_3\rangle \\ &= QFT^\dagger\left(\frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1} e^{2\pi\imath j\varphi}|j\rangle|u\rangle\right) \\ &= |\widetilde{\varphi}\rangle|u\rangle \end{aligned}$$

5- Measure the first register.

$$\rightarrow \ \widetilde{\varphi}$$

The measured value $\widetilde{\varphi}$ is an approximate value of $\varphi 2^t$. In the first register we have used t bits as shown above. Therefore, $\widetilde{\varphi}$ is the approximate value of $\varphi 2^t$ which is expressed in t bits. The number t can be determined according to the formula [1]:

$$t = n + \left\lceil \log\left(2 + \frac{1}{2\epsilon}\right)\right\rceil$$

So that the measured value $\widetilde{\varphi}$ is accurate to $\varphi 2^t$ in n bits with the probability success at least $1 - \epsilon$.

4.4.3. Order Finding

For positive integers x and N, where $x < N$, having no common factors, the *order* of x modulo N is defined to be the least positive integer r obeying the formula:

$$x^r = 1 \quad (mod \quad N)$$

Therefore, the order finding problem is to determine the integer r, in the case that the integers x and N are given. In classical computation there is no known algorithm to solve the order finding problem with polynomial resources in the sense of time and gates. We need $O(L)$ to specify the problem where $L \equiv \lceil log(N) \rceil$ is the number of bits to denote the integer N. However, by taking the Phase Estimation algorithm as the basis, quantum solution for Order Finding problem uses polynomial resources. The algorithm is mainly the phase estimation algorithm applied to the unitary operator $U_{x,N}$ defined as:

$$U_{x,N}|y\rangle \equiv |xy \ mod \ N\rangle$$

If we consider the state denoted as:

$$\begin{aligned}
|u_s\rangle &\equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{-2\pi\imath sk}{r}} |x^k \ mod \ N\rangle \\
&= \frac{1}{\sqrt{r}} \left(e^{\frac{-2\pi\imath s0}{r}} |x^0 \ mod \ N\rangle + e^{\frac{-2\pi\imath s1}{r}} |x^1 \ mod \ N\rangle + ... + e^{\frac{-2\pi\imath s(r-1)}{r}} |x^{(r-1)} \ mod \ N\rangle \right) \\
&= \frac{1}{\sqrt{r}} \left(|1 \ mod \ N\rangle + e^{\frac{-2\pi\imath s1}{r}} |x \ mod \ N\rangle + ... + e^{\frac{-2\pi\imath s(r-1)}{r}} |x^{r-1} \ mod \ N\rangle \right)
\end{aligned}$$

with a simple calculation knowing the fact that $x^r = 1 \ mod \ N$:

$$\begin{aligned}
U_{x,N}|u_s\rangle &= U_{x,N} \left(\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{-2\pi\imath sk}{r}} |x^k \ mod \ N\rangle \right) \\
&= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{-2\pi\imath sk}{r}} |x^{k+1} \ mod \ N\rangle \\
&= \frac{1}{\sqrt{r}} (e^{\frac{-2\pi\imath s0}{r}} |x^{0+1} \ mod \ N\rangle + e^{\frac{-2\pi\imath s1}{r}} |x^{1+1} \ mod \ N\rangle + ... \\
&\quad + e^{\frac{-2\pi\imath s(r-2)}{r}} |x^{(r-2)+1} \ mod \ N\rangle + e^{\frac{-2\pi\imath s(r-1)}{r}} |x^{(r-1)+1} \ mod \ N\rangle \\
&= \frac{1}{\sqrt{r}} (|x \ mod \ N\rangle + e^{\frac{-2\pi\imath s1}{r}} |x^2 \ mod \ N\rangle + ... \\
&\quad + e^{\frac{-2\pi\imath s(r-2)}{r}} |x^{(r-1)} \ mod \ N\rangle + e^{\frac{-2\pi\imath s(r-1)}{r}} |1 \ mod \ N\rangle \\
&= \frac{1}{\sqrt{r}} e^{\frac{2\pi\imath s}{r}} \left(|1 \ mod \ N\rangle + e^{\frac{-2\pi\imath s1}{r}} |x \ mod \ N\rangle + ... + e^{\frac{-2\pi\imath s(r-1)}{r}} |x^{r-1} \ mod \ N\rangle \right) \\
&= e^{\frac{2\pi\imath s}{r}} |u_s\rangle
\end{aligned}$$

we can easily understand that for $0 \leq s \leq r$, $|u_s\rangle$ are the eigenstates of the unitary transformation $U_{x,N}$ and $e^{\frac{2\pi\imath s}{r}}$ are the eigenvalues. However, in Shor's algorithm instead of using $U_{x,N}$, controlled-$U_{x,N}$ which will apply $U^j_{x,N}$ according to the value of the control register in j^{th} bit and will be denoted as $U^j_{x,N}$. By the usage of controlled unitary transformation the Order Finding procedure is shown in Figure 4.15.

1. Initialization of the first register with t qubits to the state $|0\rangle^{\otimes t}$ and the second register of L qubits to the state $|1\rangle^{\otimes L}$.
2. Apply Hadamard to the first register of t qubits of $|0\rangle^{\otimes t}$ state.
3. Apply controlled-$U^j_{x,N}$.
4. Apply the inverse Fourier transform to the first register.
5. Measure the first register.
6. Apply the Continued Fractions Algorithm. Each denominator less than N is the candidate for the order r, and this can be checked directly using modular exponentiation.

Figure 4.15. Order Finding Algorithm

Another property of the states $|u_s\rangle$ is:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle \quad = \quad |1\rangle$$

Proof:

$$\begin{aligned} \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{-2\pi\imath sk}{r}} |x^k \ mod \ N\rangle \\ &= \frac{1}{r} \sum_{k=0}^{r-1} \sum_{s=0}^{r-1} e^{\frac{-2\pi\imath sk}{r}} |x^k \ mod \ N\rangle \end{aligned} \tag{4.7}$$

With a simple analysis it is easy to show the property of equation 4.7 by examining for two cases:

1. k=0
2. $k \neq 0$

1- For k=0 :

$$\begin{aligned} &= \frac{1}{r}\sum_{s=0}^{r-1} e^{\frac{-2\pi\imath s0}{r}}|x^0 \ mod \ N\rangle \\ &= \frac{1}{r}\sum_{s=0}^{r-1}|1\rangle \\ &= |1\rangle \end{aligned}$$

2- For $k \neq 0$:

$$\begin{aligned} &= \frac{1}{r}\sum_{s=0}^{r-1} e^{\frac{-2\pi\imath sk}{r}}|x^k \ mod \ N\rangle \\ &= 0 \end{aligned}$$

The above property will be used in one of the basic points of the algorithm. If we analyze the algorithm step by step:
1- Initialization of the first register with t qubits to the state $|0\rangle^{\otimes t}$ and the second register of L qubits to the state $|1\rangle$.

$$|\psi_1\rangle = |0\rangle^{\otimes t} \otimes |1\rangle$$

2- Apply Hadamard to the first register of t qubits of $|0\rangle^{\otimes t}$ state.

$$\begin{aligned} |\psi_2\rangle &= (H^{\otimes t} \otimes I^{\otimes L})|\psi_1\rangle \\ &= (H^{\otimes t} \otimes I^{\otimes L})|0\rangle^{\otimes t} \otimes |1\rangle \\ &= \frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle \otimes |1\rangle \end{aligned}$$

Since we know that $\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = |1\rangle$, we can put $\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle$ instead of the state in the second register. So the state can be written in terms of the eigenstates of

the operator U:

$$\begin{aligned} |\psi_2\rangle &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle \otimes |1\rangle \\ &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle \otimes \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle \\ &= \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} |j\rangle \otimes |u_s\rangle \end{aligned}$$

3- Apply controlled-$U_{x,N}^j$.

$$\begin{aligned} |\psi_3\rangle &= controlled - U_{x,N}^j \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} |j\rangle \otimes |u_s\rangle \\ &= \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} |j\rangle \otimes controlled - U_{x,N}^j |u_s\rangle \\ &= \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{\frac{2\pi\imath js}{r}} |j\rangle \otimes |u_s\rangle \end{aligned}$$

4- Apply the inverse Fourier transform to the first register.

$$\begin{aligned} |\psi_4\rangle &= \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} \frac{2\pi\imath js}{r} QFT^\dagger |j\rangle \otimes |u_s\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle \otimes |u_s\rangle \end{aligned}$$

5- Measure the first register.

The value in the first register is $(s/r)2^t$. So the problem is to be able to find out the value in the denominator. The solution to this problem is the Continued Fractions algorithm, which constructs the sixth step of the algorithm.

6- Apply the Continued Fractions Algorithm [1]. The continued fractions algorithm is explained in Appendix B. This gives us the number r in the denominator, which is also the order r of x for N.

4.4.4. Shor's Factorization Algorithm

Shor's Factorization Algorithm [1, 6, 11] brings a solution to the factorization of a given integer N basically relying on the fact that for an integer z providing the property:

$$z^2 \;=\; 1 \;(mod\ N)$$

one can apply the simple mathematical formula:

$$\begin{aligned} z^2 - 1 &= 0 \;(mod\ N) \\ (z-1)(z+1) &= 0 \;(mod\ N) \end{aligned}$$

As a result, if $z + 1 \neq 0\ mod\ N$ then it is obvious that both $(z - 1)$ and $(z + 1)$ includes factors of the integer N. So in order to find the factors the only necessary computation to do is to find the greatest common divisors of N and $(z - 1)$,N and $(z + 1)$.

If we find an integer x with an even order r for the integer N, the factorization problem for N will be probably solved. The point where the order finding algorithm comes into play is the part to find an order for the integer x.
The whole procedure of Shor's algorithm for factorization is in Figure 4.16:

When we make step by step analysis of the routine:
1- If N is even, return 2.
If the integer N is even then a factor of N is already found which is 2.
2- Check whether N is a power of an integer greater than 1.
If N is not even then we should check whether it can be denoted in the form $N = a^b$ where $a \geq 1$ and $b \geq 2$. This can be found by the polynomial classical algorithm in $O(n^3)$ where n is the number of bits to represent N [1].
3- Randomly select a positive integer x smaller than N. If their greatest common

1. If N is even, return 2.
2. Check whether N is a power of an integer greater than 1 with polynomial complexity $O(n^3)$[1].
3. Randomly select a positive integer x smaller than N. If their greatest common divisor is greater than 1, return that integer.
4. If their greatest common divisor is equal to 1, then apply the order-finding routine to the determined integers x and N to find the order r.
5. If r is even and $x^{r/2} \neq -1$ then find $gcd(x^{r/2}-1, N)$ and $gcd(x^{r/2}+1, N)$. Check for the found integers for being factors of N.

Figure 4.16. Shor's Factorization Algorithm

divisor is greater than 1, return that integer.

If we can find an integer which has no common factor with N then we can determine the order of the integer. The integer can be chosen randomly in the range $[0, N]$. Besides, finding whether the random integer x and N have factor in common has a polynomial solution with the very well known, greatest common divisor(gcd) algorithm. So what we have after this step is an integer x with the property $gcd(x, N) = 1$.

4- If their greatest common divisor is equal to 1, then apply the order-finding routine to the determined integers x and N to find the order r.

Since we know the integers x and N then by applying the order finding algorithm explained in section 10, we will be able to find an integer r such that $x^r = 1 \ mod \ N$.

5- If r is even and $x^{r/2} \neq -1$ then find $gcd(x^{r/2} - 1, N)$ and $gcd(x^{r/2} + 1, N)$. Check for the found integers for being factors of N.

If the order r is even then we can make the decomposition:

$$\begin{aligned} x^r &= 1 \ mod \ N \\ x^r - 1 &= 0 \ mod \ N \\ (x^{r/2} - 1)(x^{r/2} + 1) &= 0 \ mod \ N \end{aligned}$$

Since we accept r to be the smallest integer to be the order of x mod N then we can assume that $x^{r/2}-1 \neq 0$. As a result the only possibility to take into consideration is whether $(x^{r/2}+1) = 0$. If $(x^{r/2}+1) = 0$ then it means that instead of including a factor of N, it contains N as the factor. If $(x^{r/2}+1) \neq 0$ then finding the greatest common divisor of $(x^{r/2}+1)$ and N with $(x^{r/2}-1)$ and N, we can find the factors of the integer N.

4.5. The Hidden Subgroup Problem and Hidden Subgroup Algorithms

Before making a clear statement about Hidden Subgroup algorithms [1, 12, 17, 18, 19] there are definitions to make. These are:

Definition 1 A *subgroup*[1] K of a group G is a subset of G which forms a group under the same group multiplication operation as G.

Definition 2 For K a subgroup of the group G, the *left coset* [1] of K in G determined by $g \in G$ is the set $gK \equiv \{gk|k \in K\}$. The *right coset* is similarly defined.

Definition 3 A set of elements $g_1, g_2, ..., g_l$ [1] in a group G is said to *generate* the group G if every element of G can be written as a product of (possibly repeated) elements from the list $g_1, g_2, ..., g_l$ and is expressed as $G = \langle g_1, g_2, ..., g_l \rangle$

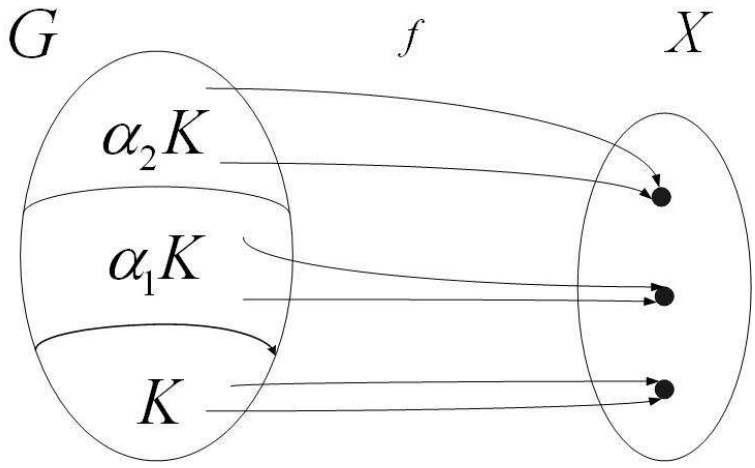

Figure 4.17. The function f is a mapping from the group G to the set X and K is the hidden subgroup we look for.

Let f be a function from a finitely generated group G to a finite set X such that f is constant on the cosets of a subgroup K of a finite index as seen in Figure 4.17. In addition, f is supposed to be distinct on each coset. In other words $f(x) = f(y)$ if and only if $x = y \diamond s$ where $\diamond$ is a certain operation like $\{+, \oplus\}$ and $s \in K$. The hidden subgroup problem is to find a generating set for K, given a way of computing f.

The quantum solution to this problem has number of operations polynomial in $log|G|$ and one use of the black box function evaluation. The procedure of the solution is in Figure 4.18.

1. Create the superpositions over the elements of the group G with Fourier Transformation.
2. Apply the quantum black box for the function f.
3. Rewrite the state of the function in the second register in the Fourier basis.
4. Apply the Inverse Fourier Transformation to the first register.
5. Measure the first register and process the measured state to gather the generating set for hidden subgroup.

Figure 4.18. Shor's Factorization Algorithm

The step by step analysis of the procedure can be found in [1, 12, 17, 18]. Deutsch's algorithm, Simon's algorithm and the Order Finding algorithm fit this general template as detailed in [1, 10].

5. VARIANTS OF HIDDEN SUBGROUP ALGORITHMS FOR GENERALIZED DOMAINS

5.1. Generalization of CKL for Input Sets of Arbitrary Size

In Ballhysa's generalization of the Deutsch-Jozsa algorithm [14], instead of the whole set of 2^n numbers, there were two sets S and S' such that their union is the whole set of 2^n numbers. The cardinality of S is accepted to be q such that $2^{n-1} < q \leq 2^n$ and it is even. We are also promised that function f is either balanced or constant on the set S and balanced on the set S'. In Ballhysa's generalization of the Deutsch-Jozsa algorithm, we are able to find out whether the function is balanced or constant on set S with one-sided error probability. We will try to apply the same convention to the algorithms we explained in sections 4.2.1 and 4.2.2 as variants of Chi, Kim and Lee Generalized Deutsch-Jozsa algorithm. The procedure will be the same as that of the Deutsch-Jozsa algorithm, except for the first Hadamard transformation to the first n qubits, we will use Equiprobable Superposition Generator, assuming that a perfect ESG is implemented as in Ballhysa's algorithm. We will first apply the generalization to CKL concerning evenly-distributed functions, then the one concerning evenly-balanced ones.

5.1.1. Generalization of CKL for Evenly-Distributed functions

As explained in section 4.2.1, the algorithm is to classify a certain function, f,to be constant or evenly-distributed. Different from the original procedure, instead of a domain with 2^n numbers, a mapping of the domain to the 2^k evenly spaced elements in the range of 2^m numbers, we will use q_n numbers in the domain such that $2^{n-1} < q < 2^n$, mapping to q_k evenly spaced numbers in the range of q_m numbers, assuming that all these numbers are even. Therefore, the equalities for $\nu = \frac{q_n}{q_k}$ and $\mu = \frac{q_m}{q_k}$ and indexing of the elements in the range $\mu j + t$ will survive. As in Ballhysa's generalization the difference will be the replacement of the first n-qubit Hadamard transformation with

n-qubit ESG. Another difference will be in the oracle. The oracle was $U|x\rangle = \omega_M^{f(x)}|x\rangle$ where $\omega_M^{f(x)} = e^{\frac{2\pi i}{M}}$ and M was 2^m however our new M here is q_m, that's why our new transformation becomes $U|x\rangle = \omega_{q_m}^{f(x)}|x\rangle$ where $\omega_{q_m}^{f(x)} = e^{\frac{2\pi i}{q_m}}$. Our assumptions will not be so different from Ballhysa's. They are:

1. q_n is greater than 2^{n-1}.
2. f is either evenly-distributed or constant on set S.

The new procedure is:

1. Initialize the qubits such that the n qubits are in state $|0\rangle^{\otimes n}$.
2. Apply ESG transformation to the n qubits.
3. Apply the oracle $U|x\rangle = \omega_{q_m}^{f(x)}|x\rangle$.
4. Apply Hadamard to all of the n qubits.
5. Measure the n-qubit state. If the measured value is different than zero, then claim that the function f is evenly-distributed. If the measured value is zero then f is constant.

Figure 5.1. Generalization of CKL in generalized domains for Evenly-Distributed functions

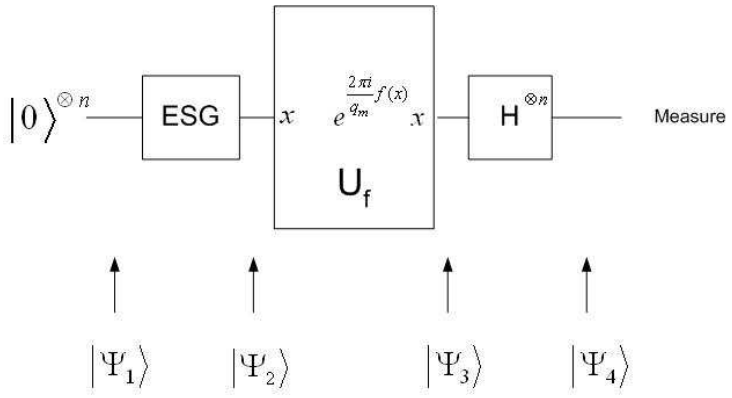

Figure 5.2. Quantum circuit for the generalization of CKL for Evenly-Distributed functions

The step by step analysis of the algorithm:

1-Initialize the qubits such that first n qubits are in state $|0\rangle^{\otimes n}$.

$$|\psi_1\rangle = |0\rangle^{\otimes n}$$

2- Apply ESG transformation to the n qubits.

$$\begin{aligned} |\psi_2\rangle &= ESG|\psi_1\rangle \\ &= \frac{1}{\sqrt{q_n}} \sum_{x \in S} |x\rangle \end{aligned}$$

2- Apply the oracle $U|x\rangle = \omega_{q_m}^{f(x)}|x\rangle$.

$$\begin{aligned} |\psi_3\rangle &= U|\psi_2\rangle \\ &= U\frac{1}{\sqrt{q_n}} \sum_{x \in S} |x\rangle \\ &= \frac{1}{\sqrt{q_n}} \sum_{x \in S} \omega_{q_m}^{f(x)}|x\rangle \end{aligned}$$

3- Apply Hadamard to all of the n qubits.

$$\begin{aligned} |\psi_4\rangle &= H^{\otimes n}|\psi_3\rangle \\ &= H^{\otimes n}\frac{1}{\sqrt{q_n}} \sum_{x \in S} \omega_{q_m}^{f(x)}|x\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{z=0}^{2^n-1} \left(\frac{1}{\sqrt{q_n}} \sum_{x \in S} (-1)^{x \cdot z} \omega_{q_m}^{f(x)} \right) |z\rangle \end{aligned} \tag{5.1}$$

4- Measure the n qubit state. Just like in the original generalization algorithm if we analyze the function f for the cases:

1. f is constant and the measured state $|z\rangle = |0\rangle$
2. f is evenly-distributed and the measured state $|z\rangle = |0\rangle$

The amplitude, S_z, of any measured state $|z\rangle$ is, as seen from the formula in 5.1 is $\frac{1}{\sqrt{2^n q_n}} \left(\sum_{x \in S} (-1)^{x \cdot z} \omega_{q_m}^{f(x)} \right)$.

For f is constant, $f(x) = f(0)$, and $|z\rangle = |0\rangle$ the amplitude of the state becomes:

$$\begin{aligned} S_0 &= \frac{1}{\sqrt{2^n q_n}} \left(\sum_{x \in S} \omega_{q_m}^{f(0)} \right) \\ &= \frac{\omega_{q_m}^{f(0)}}{\sqrt{2^n q_n}} \left(\sum_{x \in S} 1 \right) \\ &= \frac{\omega_{q_m}^{f(0)}}{\sqrt{2^n q_n}} q_n \\ &= \omega_{q_m}^{f(0)} \sqrt{\frac{q_n}{2^n}} \end{aligned}$$

So the probability of correctly observing the state $|0\rangle$ when the function f is constant is:

$$\begin{aligned} P_0 &= |S_0|^2 \\ &= |\omega_{q_m}^{f(0)} \sqrt{\frac{q_n}{2^n}}|^2 \end{aligned}$$

Since

$$\begin{aligned} |\omega_{q_m}^{f(0)}|^2 &= |e^{\frac{2\pi i f(0)}{q_m}}|^2 \\ &= 1 \end{aligned}$$

Therefore,

$$P_0 = \frac{q_n}{2^n}$$

When we analyze the amplitude of the state $|z\rangle = |0\rangle$ when the function f is evenly-distributed:

$$\begin{aligned} S_0 &= \frac{1}{\sqrt{2^n q_n}} \left(\sum_{x \in S} \omega_{q_m}^{f(x)} \right) \\ &= \frac{1}{\sqrt{2^n (\nu q_k)}} \nu \left(\sum_{j=0}^{q_k - 1} \omega_{q_m}^{\mu j + t} \right) \end{aligned}$$

$$
\begin{aligned}
&= \sqrt{\frac{\nu}{2^n q_k}} \left(\sum_{j=0}^{q_k-1} e^{\frac{2\pi i}{q_m}(\mu j + t)} \right) \\
&= \sqrt{\frac{\nu}{2^n q_k}} \left(\sum_{j=0}^{q_k-1} e^{\frac{2\pi i}{\mu q_k}(\mu j + t)} \right) \\
&= \sqrt{\frac{\nu}{2^n q_k}} \left(\sum_{j=0}^{q_k-1} e^{\frac{2\pi i j}{q_k}} e^{\frac{2\pi i}{q_m} t} \right) \\
&= \sqrt{\frac{\nu}{2^n q_k}} \omega_{q_m}^t \left(\sum_{j=0}^{q_k-1} e^{\frac{2\pi i j}{q_k}} \right) \\
&= 0
\end{aligned}
$$

The probability of observing the state $|0\rangle$ when the function f is evenly-distributed is 0. As in Ballhysa's generalization of the Deutsch-Jozsa algorithm, this generalization has the property of "one-sided error", such that the probability of reporting that the function is constant correctly is $\frac{q_n}{2^n}$, since we know that $q_n > 2^{n-1}$ then the error probability is less than $\frac{1}{2}$.

5.1.2. Generalization of CKL for Evenly-Balanced functions

In this section, we will apply the same generalization background to the algorithm explained in Section 4.2.3. In this case we try to understand whether a function f is constant or evenly-balanced when there is no other possibility for f. The applied procedure will almost be the same except for the first n-qubit transformation with Hadamard gates, we will be using ESG again. In addition, we will again be talking about sets with cardinality that are not powers of 2. The assumptions will almost be the same of section 5.1.1 except for the type of the function. This time we will try to understand whether the function is evenly-balanced or not. The assumptions are:

1. q_n is even and greater than 2^{n-1}
2. f is either evenly-balanced or constant on the set S.

The algorithm is:

1. Initialize the n-qubit control register to $|0\rangle^{\otimes}n$
2. Apply ESG transformation to the n-qubit control register.
3. Apply the oracle $U_f^{\otimes n}$ defined as $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$.
4. Apply Pauli-spin matrix corresponding to phase-flip operator $\sigma_z^{\otimes m}$ to the m-qubit auxiliary register.
5. Apply the oracle U_f to the control register again.
6. Apply phase-flip operator to the auxiliary register once more.
7. Apply Hadamard transformation to the control register once more.
8. Measure the first register, if the value is different than zero claim that f is evenly-balanced, if the measured value is zero then f is constant.

Figure 5.3. Algorithm for Chi,Kim and Lee generalization of Deutsch-Jozsa algorithm in generalized domains

The step by step analysis of the algorithm is as follows:

1- Initialize the n-qubit control register to $|0\rangle^{\otimes n}$. We also assume that the state of the m qubit auxiliary register can be stated as :

$$|\psi\rangle = \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle$$

As a result, our initial state becomes:

$$\begin{aligned} |\psi_1\rangle &= |0\rangle^{\otimes n} \otimes |\psi\rangle \\ &= |0\rangle^{\otimes n} \otimes \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \end{aligned}$$

2- Apply ESG transformation to the n-qubit control register.

$$\begin{aligned} |\psi_2\rangle &= ESG \otimes I^{\otimes m}|\psi_1\rangle \\ &= ESG \otimes I^{\otimes m}\left(|0\rangle^{\otimes n} \otimes |\psi\rangle\right) \\ &= \frac{1}{\sqrt{q_n}} \sum_{x \in S} |x\rangle \otimes \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \end{aligned}$$

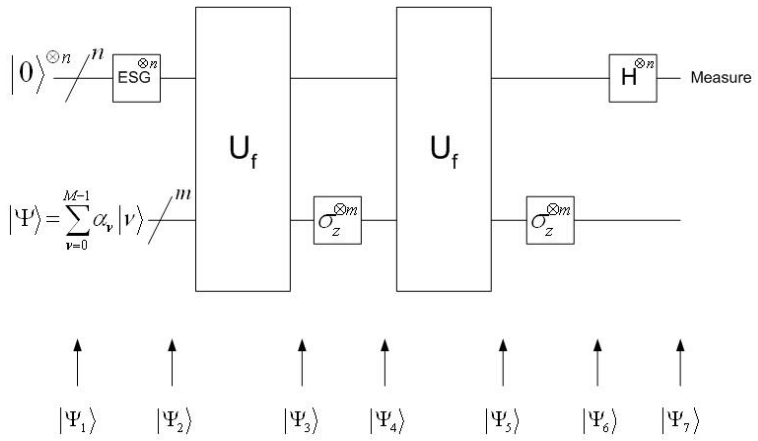

Figure 5.4. Circuit of the generalized Chi,Kim and Lee generalization of Deutsch's algorithm for Evenly-Balanced functions

3- Apply the oracle U_f defined as $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$.

$$\begin{aligned} |\psi_3\rangle &= U_f|\psi_2\rangle \\ &= \frac{1}{\sqrt{q_n}} \sum_{x\in S} |x\rangle \otimes \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} (|\nu_j\rangle \oplus f(x)_j) \end{aligned}$$

4- Apply Pauli-spin matrix corresponding to phase-flip operator $\sigma_z^{\otimes m}$ to the m-qubit auxiliary register.

$$\begin{aligned} |\psi_4\rangle &= \left(I^{\otimes n} \otimes \sigma_z^{\otimes m}\right) |\psi_3\rangle \\ &= \left(I^{\otimes n} \otimes \sigma_z^{\otimes m}\right) \frac{1}{\sqrt{q_n}} \sum_{x\in S} |x\rangle \otimes \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j \oplus f(x)_j\rangle \\ &= \frac{1}{\sqrt{q_n}} \sum_{x\in S} |x\rangle \otimes \sum_{\nu=0}^{M-1} (-1)^{\sum_{j=0}^{m-1}(v_j + f(x)_j)} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j \oplus f(x)_j\rangle \\ &= \frac{1}{\sqrt{q_n}} \sum_{x\in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes \sum_{\nu=0}^{M-1} (-1)^{\sum_{j=0}^{m-1} v_j} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j \oplus f(x)_j\rangle \end{aligned}$$

5- Apply the oracle $U_f^{\otimes n}$ to the control register again.

$$\begin{aligned} |\psi_5\rangle &= U_f|\psi_4\rangle \\ &= U_f \frac{1}{\sqrt{q_n}} \sum_{x\in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes \sum_{\nu=0}^{M-1} (-1)^{\sum_{j=0}^{m-1} v_j} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j \oplus f(x)_j\rangle \end{aligned}$$

$$= \frac{1}{\sqrt{q_n}} \sum_{x \in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes \sum_{\nu=0}^{M-1} (-1)^{\sum_{j=0}^{m-1} v_j} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j \oplus f(x)_j \oplus f(x)_j\rangle$$
$$= \frac{1}{\sqrt{q_n}} \sum_{x \in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes \sum_{\nu=0}^{M-1} (-1)^{\sum_{j=0}^{m-1} v_j} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle$$

6- Apply phase-flip operator to the auxiliary register once more.

$$\begin{aligned}
|\psi_6\rangle &= \left(I^{\otimes n} \otimes \sigma_z^{\otimes m}\right) |\psi_5\rangle \\
&= \left(I^{\otimes n} \otimes \sigma_z^{\otimes m}\right) \frac{1}{\sqrt{q_n}} \sum_{x \in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes \sum_{\nu=0}^{M-1} (-1)^{\sum_{j=0}^{m-1} v_j} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \\
&= \frac{1}{\sqrt{q_n}} \sum_{x \in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes \sum_{\nu=0}^{M-1} (-1)^{2\sum_{j=0}^{m-1} v_j} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \\
&= \frac{1}{\sqrt{q_n}} \sum_{x \in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes \sum_{\nu=0}^{M-1} \alpha_\nu \bigotimes_{j=0}^{m-1} |\nu_j\rangle \\
&= \frac{1}{\sqrt{q_n}} \sum_{x \in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes |\psi\rangle
\end{aligned}$$

7- Apply Hadamard transformation to the control register once more.

$$\begin{aligned}
|\psi_7\rangle &= \left(H^{\otimes n} \otimes I^{\otimes m}\right) |\psi_6\rangle \\
&= \left(H^{\otimes n} \otimes I^{\otimes m}\right) \frac{1}{\sqrt{q_n}} \sum_{x \in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} |x\rangle \otimes |\psi\rangle \\
&= \sum_{z=0}^{2^n-1} \left\{ \frac{1}{\sqrt{2^n q_n}} \sum_{x \in S} (-1)^{x \cdot z} (-1)^{\sum_{j=0}^{m-1} f(x)_j} \right\} |z\rangle \otimes |\psi\rangle
\end{aligned}$$

The amplitude of any measurable state is:

$$S_z = \frac{1}{\sqrt{2^n q_n}} \sum_{x \in S} (-1)^{x \cdot z} (-1)^{\sum_{j=0}^{m-1} f(x)_j}$$

As a result, for the state $|z\rangle = |0\rangle$ the amplitude becomes:

$$S_0 = \frac{1}{\sqrt{2^n q_n}} \sum_{x \in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} \tag{5.2}$$

There are two cases to consider:

1. f is constant
2. f is evenly-balanced

For f is constant, $f(x) = f(0)$:

$$\begin{aligned} S_0 &= \frac{1}{\sqrt{2^n q_n}} \sum_{x \in S} (-1)^{\sum_{j=0}^{m-1} f(0)_j} \\ &= (-1)^{\sum_{j=0}^{m-1} f(0)_j} \frac{q_n}{\sqrt{2^n q_n}} \\ &= (-1)^{\sum_{j=0}^{m-1} f(0)_j} \sqrt{\frac{q_n}{2^n}} \end{aligned}$$

So the probability to observe the state $|0\rangle$ is:

$$\begin{aligned} P_0 &= |S_0|^2 \\ &= \left| (-1)^{\sum_{j=0}^{m-1} f(0)_j} \sqrt{\frac{q_n}{2^n}} \right|^2 \\ &= \frac{q_n}{2^n} \end{aligned}$$

If we consider the amplitude of the state $|z\rangle = |0\rangle$ when the function is evenly-balanced:

$$\begin{aligned} S_0 &= \frac{1}{\sqrt{2^n q_n}} \sum_{x \in S} (-1)^{\sum_{j=0}^{m-1} f(x)_j} \\ &= 0 \end{aligned}$$

The "one-sided error" property exists here since we know that if we measure the value $|0\rangle$ f is definitely constant. When we measure a value different than $|0\rangle$, we will claim that f is evenly-balanced.

5.2. Generalization of Simon's Algorithm

As we generalized the previous algorithms, we can generalize Simon's algorithm in the same manner and the success probability does not differ from the original algorithm. The procedure will be the same as the original Simon's algorithm with the

same difference of using ESG instead of Hadamard as in all other generalizations. As in other generalizations our assumptions are:

1. q is even and $q > 2^{n-1}$
2. f is either 1-to-1 with a trivial $s = 0^{\otimes n}$ or 2-to-1 with a non-trivial string s.

The algorithm is:

1. Initialize totally $n+m$ qubits, n qubits for the first register and m qubits for the second one, to the state $|0\rangle^{\otimes n+m}$
2. Apply ESG transformation to the n-qubit first register.
3. Apply oracle U_f defined as $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$.
4. Apply Hadamard transformation to the first register.
5. Measure the first register as a row to the matrix of step 6.
6. Repeat the first five steps for $n-1$ times to constitute an $n-1 \times n$ non-singular matrix with output strings as row vectors.
7. Solve the matrix problem $A_{n-1\times n}s_{n\times 1} = 0$ by using Gaussian elimination.
8. Check for the property of the function f by evaluating the values for $f(0)$ and $f(s)$.

Figure 5.5. Simon's algorithm in generalized domains

When we analyze the algorithm step by step:

1- Initialize totally $n + m$ qubits, n qubits for the first register and m qubits for the second one, to the state $|0\rangle^{\otimes n+m}$.

$$|\psi_1\rangle = |0\rangle^{\otimes(n+m)}$$

2- Apply ESG transformation to the n qubit first register.

$$\begin{aligned} |\psi_2\rangle &= (ESG \otimes I^{\otimes m})|\psi_1\rangle \\ &= \frac{1}{\sqrt{q}} \sum_{x\in S} |x\rangle \otimes |0\rangle^{\otimes m} \end{aligned}$$

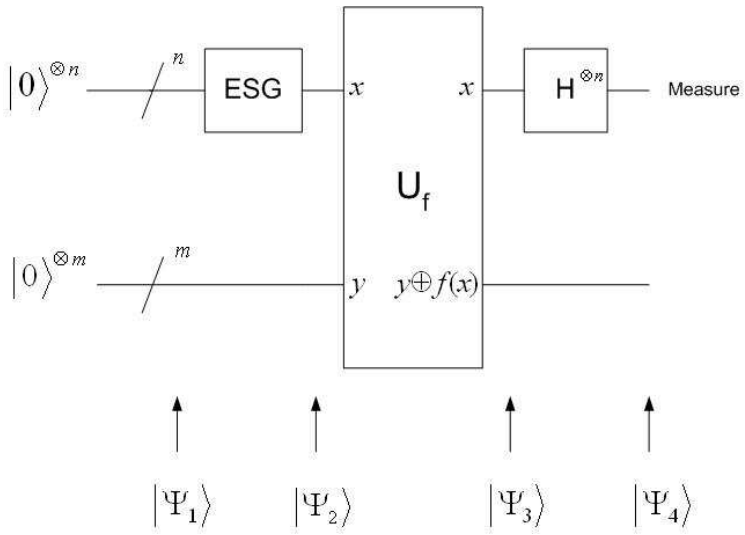

Figure 5.6. Quantum Circuit for first four steps of Generalized Simon's algorithm

3- Apply oracle U_f mentioned in Deutsch's problem.

$$\begin{aligned} |\psi_3\rangle &= U_f|\psi_2\rangle \\ &= U_f \frac{1}{\sqrt{q}} \sum_{x\in S} |x\rangle \otimes |0\rangle^{\otimes m} \\ &= \frac{1}{\sqrt{q}} \sum_{x\in S} |x\rangle \otimes |f(x)\rangle^{\otimes m} \end{aligned}$$

4- Apply Hadamard transformation to the first register.

$$\begin{aligned} |\psi_4\rangle &= (H^{\otimes n} \otimes I^{\otimes m})|\psi_3\rangle \\ &= (H^{\otimes n} \otimes I^{\otimes m})(\frac{1}{\sqrt{q}} \sum_{x\in S} |x\rangle \otimes |f(x)\rangle^{\otimes m}) \\ &= \frac{1}{\sqrt{2^n q}} \sum_{z=0}^{2^n-1} \sum_{x\in S} (-1)^{x\cdot z} |z\rangle \otimes |f(x)\rangle^{\otimes m} \end{aligned}$$

When we check for the amplitude of any state $|z\rangle \otimes |f(x)\rangle$ then if the function is 2-to-1, meaning that there exists string s such that $\forall x \neq y$:

$$f(x) = f(x') \quad \Leftrightarrow \quad x' = x \oplus s \tag{5.3}$$

Then, the amplitude of any state $|z\rangle \otimes |f(x)\rangle$ becomes as shown in the original Simon's algorithm:

$$S_z = \frac{1}{\sqrt{2^n q}} \sum_{x \in S} \left((-1)^{x \cdot z} + (-1)^{(x \oplus s) \cdot z} \right) \tag{5.4}$$

We already know that $(x \oplus s) \cdot z = x \cdot z + s \cdot z (mod 2)$. As a result,if $s \cdot z \neq 0$, then the amplitude will be:

$$\begin{aligned} S_z &= \frac{1}{\sqrt{2^n q}} \left((-1)^{x \cdot z} + (-1)^{(x \oplus s) \cdot z} \right) \\ &= \frac{1}{\sqrt{2^n q}} \left((-1)^{x \cdot z} + (-1)^{x \cdot z + s \cdot z} \right) \\ &= \frac{1}{\sqrt{2^n q}} \left((-1)^{x \cdot z} + (-1)^{x \cdot z + 1} \right) \\ &= 0 \end{aligned}$$

Therefore, the probability that we will be observing the true string t with the property $t \cdot s = 0$ is 1.
Measure the first register as a row to the matrix of step 6.
Measured string at the end of the first five steps will be a row of length n in the matrix of step 6.
6- Repeat the first five procedures for $n-1$ times to constitute an $n-1 \times n$ non-singular matrix with output strings as row vectors.

The analysis of the algorithm will not differ from the one in section 4.3, since we will be measuring z out of 2^{n-1} possible states. The issue is that we are still able to observe z with the property $s \cdot z = 0$. Our new domain puts limitation for the number of x's, not for the set z belongs to. Although the number of x's decreased, the probability that we will measure z's with the property $s \cdot z = 0$ remains the same which is 1. Therefore the analysis of section 4.3 does not change, we will repeat the first four procedure for $n-1$ times to hopefully get $n-1$ independent vectors out of 2^{n-1} possibilities. As a result, we are able to determine whether function f is 2-to-1 or 1-to-1 with probability greater than 1/4.

6. CONCLUSIONS

In this work, we have discussed the effects of generalized domains consisting of number of elements that are not necessarily powers of two on three algorithms. The first two were Chi, Kim and Lee generalizations of the Deutsch-Jozsa algorithm for functions which are evenly-distributed and evenly-balanced [13]. In these algorithms, the generalized domain algorithm worked not perfectly but with "one-sided error" property. The third algorithm was Simon's algorithm [5]. In Simon's algorithm, the generalized algorithm worked perfectly such that the error probability of original Simon's algorithm have not changed.

Our current aim is to analyze the error probabilities of these and other hidden subgroup algorithms for the cases where the equiprobable superposition generator works imperfectly as in the KSV and G algorithms and possible newly created equiprobable superposition generators.

APPENDIX A: GAUSSIAN ELIMINATION ON BINARY NUMBERS

The general Gaussian algorithm works on the equation $Ax = b$ where A is $n \times n$ matrix, x is an unknown $n \times 1$ vector and b is a $n \times 1$ vector. The task is to find the vector x. The application of Gaussian algorithm accepts the assumption that the rows of the matrix A are independent from each other, so that it is possible to come up with an upper triangular matrix by applying elementary row operations on A. The mathematical perspective is as follows:

$$\begin{array}{cccc} A & & x & = & b \end{array}$$
$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Before applying elementary row operations we exchange the vector x with vector b and treat A and b as if they form a new single matrix. During row operations vector x is also applied the same row operations in itself. After the application of row operations we end up with:

$$\left[\begin{array}{cccc|c} a'_{11} & a'_{12} & \dots & a'_{1n} & b'_1 \\ 0 & a'_{22} & \dots & a'_{2n} & b'_2 \\ 0 & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & a'_{nn} & b'_n \end{array}\right] \qquad \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{bmatrix}$$

We exchange the places of the vectors x and b once more, to be able to make matrix computations:

$$\left[\begin{array}{cccc|c} a'_{11} & a'_{12} & \dots & a'_{1n} & \\ 0 & a'_{22} & \dots & a'_{2n} & \\ 0 & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & a'_{nn} & \end{array}\right] \left[\begin{array}{c} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{array}\right] = \left[\begin{array}{c} b'_1 \\ b'_2 \\ \vdots \\ b'_n \end{array}\right] \tag{A.1}$$

This results in n equations. Dissolving each x'_i one by one easily, using backsubstitution, we can compute all the x_i values. The binary application of the Gaussian elimination will almost be the same, although note that in our case there are $n-1$ equations. Instead of multiplying rows with coefficients to get an upper triangular matrix, we use the fact that in binary arithmetic, 0+1=1;0+0=0 and 1+1=0.

We shall first start with why it is enough for us to have $n-1 \times n$ matrix whose rows are independent from each other. In binary case, having a single 1 or double 1's in a row are the basis cases. Suppose that there is a row with a single one corresponding to s_0 in our string s. This will mean that s_0 is zero. If we have two 1's in a row corresponding to s_0 and s_2 then this will mean that $s_0 + s_2 = 0$. Since our domain is binary numbers we can conclude that $s_0 = s_2$. This allows us to replace all s_2's with s_0's or vice a versa. In original Gaussian elimination we have expressed that we end up with equation A.1 with a single element in the last row. In the binary case it is enough for us to have two elements in the last row as in the $(n-1)^{th}$ row in equation A.1. This allows us to conclude that $n-1$ rows will be enough to find the string s in binary Gaussian algorithm. As mentioned above, our initial aim is to find rows with a single 1 or double 1's.

When explaining the algorithm, an example will serve us the best as follows: Suppose that we have a 5×6 matrix A such that:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

In Simon's algorithm our vector b clearly equals:

$$b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Since the issue is to find the string s such that $As = 0$.

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Our first aim is to find rows consisting of one or two 1's. If our matrix does not have such rows then our initial aim will be to modify our matrix by elementary row

operations as in original Gaussian elimination process. For our example:

$$\begin{array}{cccccc} s_0 & s_1 & s_2 & s_3 & s_4 & s_5 \\ 1 & 1 & 0 & 0 & 1 & 0 \quad \rightarrow \quad 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \quad \rightarrow \quad 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \quad \rightarrow \quad 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \quad \rightarrow \quad 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \quad \rightarrow \quad 0 \end{array}$$

When we look at our matrix, we see that there is no row with a single 1 or two 1's. As a result we will start our row operations. We can target the first row at the beginning as using it for the addition operation to leave a single or two 1's in the s_0 column as follows:

$$\begin{array}{cccccc} s_0 & s_1 & s_2 & s_3 & s_4 & s_5 \\ 1 & 1 & 0 & 0 & 1 & 0 \quad \rightarrow \quad 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \quad \rightarrow \quad 0 \\ 1+1=0 & 0+1=1 & 1+0=1 & 0+0=0 & 0+1=1 & 1+0=1 \quad \rightarrow \quad 0 \\ 1+1=0 & 1+1=0 & 1+0=1 & 1+0=1 & 0+1=1 & 1+0=1 \quad \rightarrow \quad 0 \\ 1+1=0 & 1+1=0 & 0+0=0 & 1+0=1 & 0+1=1 & 1+0=1 \quad \rightarrow \quad 0 \end{array}$$

When we look at the rows there are still no rows with single 1 or two 1's. As a result, for this time we target the second column corresponding to s_1 by adding the second row to the other ones with a 1 in the same column.

$$\begin{array}{cccccc} s_0 & s_1 & s_2 & s_3 & s_4 & s_5 \\ 1 & 1 & 0 & 0 & 1 & 0 \quad \rightarrow \quad 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \quad \rightarrow \quad 0 \\ 0 & 1+1=0 & 1+1=0 & 0+1=1 & 1+0=1 & 1+1=0 \quad \rightarrow \quad 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \quad \rightarrow \quad 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \quad \rightarrow \quad 0 \end{array}$$

After this second elimination, the equation becomes:

$$\begin{array}{cccccccc} s_0 & s_1 & s_2 & s_3 & s_4 & s_5 & & \\ 1 & 1 & 0 & 0 & 1 & 0 & \rightarrow & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & \rightarrow & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & \rightarrow & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & \rightarrow & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & \rightarrow & 0 \end{array}$$

When we search for the single 1 or two 1's in a row, we see that in the third row there are two 1's corresponding to columns of s_3 and s_4. We can conclude that $s_3 + s_4 = 0$, meaning that $s_3 = s_4$. With this knowledge I can put s_4 in place of s_3 in other words we can add the column of s_3 and s_4 and name it $s_{3,4}$.

$$\begin{array}{ccccccc} s_0 & s_1 & s_2 & s_{3,4} & s_5 & & \\ 1 & 1 & 0 & 1 & 0 & \rightarrow & 0 \\ 0 & 1 & 1 & 1 & 1 & \rightarrow & 0 \\ 0 & 0 & 0 & 0 & 0 & \rightarrow & 0 \\ 0 & 0 & 1 & 0 & 1 & \rightarrow & 0 \\ 0 & 0 & 0 & 0 & 1 & \rightarrow & 0 \end{array}$$

When we look at the matrix we see that there are all zero's in a row and we delete them, besides in the fifth row above, there exists a single 1, meaning that $s_5 = 0$. We can replace s_5 with zeros and delete the corresponding row and column.

$$\begin{array}{cccccc} s_0 & s_1 & s_2 & s_{3,4} & & \\ 1 & 1 & 0 & 1 & \rightarrow & 0 \\ 0 & 1 & 1 & 1 & \rightarrow & 0 \\ 0 & 0 & 1 & 0 & \rightarrow & 0 \end{array}$$

We see that the last row has only one 1, which tells us that the column corresponding to that single 1, s_2 is zero. As we did before, we can replace 0's in place of s_2 and delete corresponding rows and columns.

$$\begin{array}{cccc} s_0 & s_1 & s_{3,4} & \\ 1 & 1 & 1 & \rightarrow \ 0 \\ 0 & 1 & 1 & \rightarrow \ 0 \end{array}$$

Since the last row has two 1's, we can say that corresponding columns s_1 and $s_{3,4}$ are equal as $s_1 = s_{3,4}$. We add the two columns and rename it as $s_{1,3,4}$.

$$\begin{array}{ccc} s_0 & s_{1,3,4} & \\ 1 & 0 & \rightarrow \ 0 \\ 0 & 0 & \rightarrow \ 0 \end{array}$$

In the last equation, since s_0 has a single 1 then we can conclude that $s_0 = 0$. Finally what we gathered is $s_0 = s_2 = s_5 = 0$ and $s_1 = s_3 = s_4$. Since we look for a non-trivial solution, we say that $s_1 = s_3 = s_4 = 1$. As a result our string becomes $s = [0\ 1\ 0\ 1\ 1\ 0]$.

As seen in the example, we make search of $n-1$ rows column by column for n elements. So there are two important steps that will be applied to each row. We first make control of the rows with a single or double 1. This control step takes $O(n^2)$ steps since we consider all of the elements in an $n-1 \times n$ matrix. The second step is the addition operations in case the control step returns that there is no row with a single or double 1. In this case starting from the first row checking for a column with a 1, we try to vanish the other 1's in the other rows of the same column which takes $O(n)$ step for one row. For all rows it takes $O(n^2)$. After each step we put our control step to make the necessary replacements. This reduces complexity of the algorithm since it deletes some of the columns and rows in most of the cases, however in the worst case

we repeat the control step of $O(n^2)$ complexity, for each row, in other words for $n-1$ times. Therefore, our complexity becomes $O(n^3)$ which is polynomial.

APPENDIX B: THE CONTINUED FRACTIONS ALGORITHM

Continued fractions is a very essential way to express rational numbers in terms of integers. Continued fractions expression is of the form:

$$r = a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{\ldots + \cfrac{b_{M-1}}{a_M}}}}$$

However, with all $b_i = 1$, continued fractions become *simple continued fractions* and is expressed as:

$$r = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ldots + \cfrac{1}{a_M}}}}$$

A more convenient notation to express simple continued fractions is $r = [a_0, a_1, ..., a_M]$, assuming that all $a_i > 0$. With an example we can make a clear description of the continued fractions by applying the algorithm on the fractional number $\frac{3796}{1387}$ as follows:

$$\begin{aligned} \frac{3796}{1387} &= 2 + \frac{1022}{1387} \\ &= 2 + \frac{1}{\frac{1387}{1022}} \end{aligned}$$

The key step of the algorithm to keep dividing the smaller number to the larger one till an integer is seen.

$$\begin{aligned} \frac{3796}{1387} &= 2 + \frac{1}{\frac{1387}{1022}} \\ &= 2 + \frac{1}{1 + \frac{365}{1022}} \end{aligned}$$

$$
\begin{aligned}
&= 2 + \frac{1}{1 + \frac{1}{\frac{1022}{365}}} \\
&= 2 + \frac{1}{1 + \frac{1}{2 + \frac{292}{365}}} \\
&= 2 + \frac{1}{1 + \frac{1}{2 + \frac{1}{\frac{365}{292}}}} \\
&= 2 + \frac{1}{1 + \frac{1}{2 + \frac{1}{1 + \frac{73}{292}}}} \\
&= 2 + \frac{1}{1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{\frac{292}{73}}}}} \\
&= 2 + \frac{1}{1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{4}}}}
\end{aligned}
$$

In short, continued fractions expression of the real number becomes:

$$\frac{3796}{1387} = [2, 1, 2, 1, 4]$$

It is clear that the algorithm will terminate after a finite number of division for any rational number, since the denominators as in the sample (3796,1022,365,292,73,1) are strictly decreasing. For a rational number in the form $\varphi = s/r$ where s and r are n bit integers, the continued fractions algorithm will be computed in $O(n^3)$ operations consisting of $O(n)$ steps for dividing and inverting using $O(n^2)$ gates for elementary arithmetic.

Longer and longer prefixes of $[2, 1, 2, 1, 4]$ are better and better approximations to the original fraction. In order-finding, since we know x, N and a range for r, we can extend the continued fraction and check whether the newest denominator is r iteratively. The fraction we must start with, is the number we read in the first register over 2^t.

REFERENCES

1. Nielsen, M. A. and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

2. Deutsch, D., "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer", *Proceedings of Royal Society of London*, Vol. A 400, pp. 97-117, 1985.

3. Bernstein, D. J. and U. Vazirani, "Quantum Complexity Theory". *SIAM Journal of Computing*, Vol. 26, No.5, pp. 1411-1473, 1997.

4. Deutsch, D. and R. Jozsa, "Rapid Solution of Problems by Quantum Computation" *Proceedings of Royal Society of London*, Vol. A 439, pp. 439-553 , 1992.

5. Simon D., "On the Power of Quantum Computation", *SIAM J. on Computing*, Vol. 26, pp. 1474-1483, 1997.

6. Shor, P. W., "Algorithms for Quantum Computing: Discrete Logarithm and Factoring", *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pp. 124-134, 1994.

7. Grover, L., "A Fast Quantum Mechanical Algorithm for Database Search", *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pp. 212-219, 1996.

8. Childs, A.M., R. Cleve, E. Deotto, E. Farhi, S. Gutmann and D.A. Spielman, "Exponential Algorithmic Speedup by a Quantum Walk", http://arxiv.org/pdf/quant-ph/0209131, 2002.

9. Shor, P. W., "Progress in Quantum Algorithms",http://www-math.mit.edu/ shor/papers/Progress.pdf, 2005.

10. Gruska J.,*Quantum Computing*, McGraw-Hill,London, 1999.

11. Hardy Y. and W.-H. Steeb, *Classical and Quantum Computing with C++ and Java Simulations*, Birkhäuser Verlag, Basel 2001.

12. Cleve, R., A. Ekert, C. Macchiavello and M. Mosca, "Quantum Algorithms Revisited", *Proceedings of Royal London Society*, Vol. A 454, pp. 339-354, 1998.

13. Chi, D. P., J. Kim and S. Lee, "Initialization-free Generalized Deutsch-Jozsa Algorithm", *Journal of Physics A: Mathematical and General*, Vol. 34, pp. 5251-5258, 2001.

14. Ballhysa, E. and A. C. Cem Say, "Generating equiprobable superpositions of arbitrary sets for a new generalization of the Deutsch-Jozsa algorithm", *Lecture Notes in Computer Science Vol. 3280, (Proceedings of the Fourteenth International Symposium on Computer and Information Sciences)*, pp. 966-975, 2004.

15. Ballhysa E., "A Generalization of the Deutsch-Jozsa Algorithm and the Development of a Quantum Programming Infrastructure",M.S. Thesis, Boğaziçi University, 2004.

16. Brassard G. and P. Høyer, "An Exact Quantum Polynomial-Time Algorithm for Simon's Problem", *ISTCS*, pp. 12-23, 1997.

17. Mosca M. and A.Ekert, "The Hidden Subgroup Problem and Eigenvalue Estimation on a Quantum Computer", *Lecture Notes in Computer Science*, Vol. 1509, pp. 174-188, 1999.

18. Meglicki Z., *Introduction to Computing*, http://beige.ucs.indiana.edu/M743/M743.pdf.

19. Jozsa R., "Quantum Algorithms and Fourier Transform", *Proceedings of Royal Society of London*, Vol. A 454, pp. 323-337, 1998.