QUANTUM ANALOGUES OF CLASSICAL OPTIMIZATION ALGORITHMS

Kelvin Mpofu

School of Chemistry and Physics

University of KwaZulu-Natal

A thesis submitted in fulfillment of the requirements for the degree of

Master of Science

December, 2017

Abstract

This thesis explores the quantum analogues of algorithms used in mathematical optimization. The thesis focuses primarily on the iterative gradient search algorithm (algorithm for finding the minimum or maximum of a function) and the Newton-Raphson algorithm. The thesis introduces a new quantum gradient algorithm suggested by Professor Thomas Konrad and colleagues and a quantum analogue of the Newton-Raphson Method, a method for finding approximations to the roots or zeroes of a real-valued function. The quantum gradient algorithm and the quantum Newton-Raphson are shown to give a polynomial speed up over their classical analogues.

Preface

The work reported in this dissertation was theoretical in nature, and was carried out in the School of Chemistry and Physics, University of KwaZulu-Natal, under the supervision of Prof. Thomas Konrad.

As the candidate's supervisor I have approved this dissertation for submission.

Prof. T.Konrad	Date
Prof. M.Tame	Date

Declaration

I, Kelvin Mpofu declare that

1. The research reported in this thesis, except where otherwise indicated, is my original research.
2. This thesis has not been submitted for any degree or examination at any other university.
3. This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
   (a) This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
   (b) Their words have been re-written but the general information attributed to them has been referenced. Where their exact words have been used, their writing has been placed inside quotation marks, and referenced.
4. This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

Signed: ______________________

Acknowledgements

First and foremost I would like to thank my supervisor Prof. Thomas Konrad, for the experience I have gained over my tenure as a master's student. This thesis would not have had been possible without him. I would also like to thank my friends and family for all the support over the course of my thesis. I could not have had been able as far as I am without their continuing support. Special thanks goes to Maria Schuld for the counsel and for some very useful suggestions which have made my thesis a better read as well as my co-supervisor Prof Mark Tame for the learning experience I had working in his lab. I would also like to thank all the scientists and researchers who have contributed and those who continue to contribute to our body of scientific knowledge. Were it not for their relentless pursuit of knowledge then certainly none of this would be possible. I would also like to thank my uncle and aunt Tapson and Rubie Rukuni for their support over the course of my studies.

Finally, I would like to thank my mother, Mary Rukuni. I would not have the courage to do what I do were it not for her support.

Contents

List of Figures

Chapter 1

Introduction and Overview

Quantum computation and quantum information are the study of the implementation of information processing and computing tasks that can be accomplished using quantum mechanical systems [1]. A quantum mechanical system is a system whose features are dictated by the rules of quantum mechanics which in itself is a mathematical framework or set of rules for the construction of physical theories that govern the behaviour of objects so small that they lie within or below the nanometre range. Examples of such objects are photons (particles of light) and atoms (building blocks of matter).

Modern quantum mechanics has become an indispensable part of science and engineering and the ideas behind the field of quantum mechanics surfaced during the early 20th century when problems in physical theories now referred to as the classical physics began to surface as the physics theories began predicting absurdities such as the existence of an ultraviolet catastrophe involving infinite energies and electrons spiralling inexorably into the atomic nucleus. In the context of computing one can view the relationship of quantum mechanics to specific physical theories as being similar to the relationship between a computer's operating system with specific applications software. Where we have that the operating system sets certain basic parameters and modes of operation, but leaves open how specific tasks are accomplished by the applications.

In 1982 the renowned physicist Richard Feynman pointed out that there were essential difficulties in simulating quantum mechanical systems on classical computers. He went onto suggest that we could build computers based on the principles of quantum mechanics in order to avoid those difficulties. This was the birth of the idea of quantum computing. In 1985 the physicist David Deutsch motivated by the question as to whether the laws of physics could be used to derive an even stronger version of the Church-Turing thesis, which states that any model of computation can be simulated on a Turing machine with a polynomial increase in number of elementary operations required [7], devised the first quantum algorithm bearing his name, the "Deutsch algorithm" for this quantum computing device proposed by Richard Feynman.

In the early 1990s researchers began investigating quantum algorithms a bit more, motivated by the fact that it was possible to use quantum computers to efficiently simulate systems that have no known efficient simulation on a classical computer. A major breakthrough came in 1994 when the mathematician Peter Shor demonstrated that two extremely important problems intractable on a classical computer could be solved on a quantum computer using an algorithm which now bears his name, "Shor's algorithm" [2]. Shor's algorithm solves the problem of finding the prime factors of an integer and the discrete logarithm problem. This discovery drove a lot of attention towards quantum computers as it further evidenced the capacity of a quantum computer over a probabilistic Turing machine by displaying an exponential speed-up. In 1995 computer scientist Lov Grover provided further evidence of the importance of quantum computers through his search algorithm, "Grover's search algorithm" [3]. The algorithm solves the problem of conducting a search through some unstructured search space(database) which he showed could be sped up on a quantum computer.

Looking at all computational devices beginning with Babbage's analytical machine all the way upto to the CRAY supercomputers we find that they work on the same fundamental principles. Some of these fundamentals extend to quantum computing even though the operations in quantum computing may be different from those in classical computing. As research in quantum computing grows people have begun trying to construct quantum algorithms which solve everyday practical problems. These include problems in numerical analysis, machine learning, artificial intelligence and mathematical optimization [4]. Some examples of these algorithms will be presented in chapter 3.

Mathematical optimization is a branch of mathematics which offers a wide range of algorithms designed to find the optimal solution to a problem. Due to the fact that optimization problems come in a wide variety there is no one universal optimization algorithm. Some problems can be solved using more than a single algorithm. Typically we determine the complexity of the algorithm and choose to use the one which offers the most efficient implementation. A few quantum algorithms which solve some of the problems in the category of mathematical optimization have been constructed, a short summary of these will be given in chapter 4 with quantum optimization algorithms.

Many researchers in quantum optimization try to find quantum algorithms that can take the place of classical machine optimization algorithms to solve a problem and show an improvement in complexity. This is the case for the gradient search algorithm proposed in this thesis.

The research described in this dissertation leads to potential quantum algorithms for solving optimization problems. The goal of the thesis is to provide a useful review of the literature on quantum optimisation which can serve as an introduction to the field and also to investigate in selected topics, how one could extend the literature by an original contribution. It also serves as a register of working principles of investigated implementations whose preliminary tests failed. It must be noted that efforts will continue to be made to improve these implementations.

1.1 Overview of the dissertation

Though the dissertation is aimed at being an introduction to the field of quantum computation especially quantum optimization it starts original research in quantum algorithm design. After offering an introduction to quantum optimization the dissertation also shows older ideas in the field of quantum optimization and goes on to propose new algorithms and some working principle whose preliminary implementation failed. The failed attempts will prove useful for the researcher who may wish to attempt a similar procedure and have been added to the appendix. They may be more useful for more advanced researchers in the field of quantum optimization looking for new ideas or simply looking at techniques which do not work. Chapter 2 introduces concepts related to the field of quantum computation including complexity, qubits, quantum Fourier transform and the Grover algorithm. Chapters 5 and 6 include the main research work as these are the proposed optimization algorithms that is the quantum gradient algorithm and quantum Newton-Raphson method .

Outline of the dissertation

In chapter 2 we introduce quantum computation as a field of study. We look at some core ideas and terminology of quantum computation. We also look at the most important algorithms in quantum computation , *i.e*, Quantum Fourier transform and the Grover algorithm. We look at these in detail and discuss applications.

In chapter 3 we introduce classical optimization algorithms as they provide the backbone of the field of quantum optimization algorithms. It is important to have an idea of the principles of classical optimization before going into quantum optimization.

Chapter 4 summarizes known quantum optimization algorithms. It must be noted that the quantum optimization algorithms proposed in this dissertation are not the first attempts at the specific problems which is why the thesis presents summaries of other known algorithms.

Chapters 5 and 6 contain our newly proposed ideas for quantum optimization algorithms.

How to read this dissertation

Due to the diverse nature of the dissertation we will give a guide to the reader so that he/she can read in the order they prefer depending on what their expectations are from the thesis. This section attempts to guide the readers in such a way that they can be more selective about reading the dissertation. If the reader requires some introductory reference to quantum computation then he/she should consider reading chapter 2.

Chapter 3 is quite vital if the reader needs an introduction to classical mathematical optimization, this section will give the reader a good impression of the type of problems solved in mathematical optimization. It should be noted that this section is not a complete review of the topic.

Chapter 4 is quite useful to read for gaining summarized knowledge about other quantum optimization algorithms which have been developed by other researchers.

Chapters 5 and 6 are for readers interested in the research work done as they contain the proposed quantum optimization algorithms. The appendices includes concepts which may be of interest to the reader if they want more information about optimization algorithms and quantum computation including a proposal for a quantum analogue of the particle swarm optimization algorithm.

Chapter 2

Quantum computation

Though there are many models used for quantum computing, for example, the adiabatic computing model (see Appendix E), quantum Turing machines and the circuit model. The more popular model is the circuit model. In this model algorithms are built from a quantum circuit of elementary quantum gates which are used to manipulate the quantum state. This is somewhat similar to the way a classical computer (processor) is built from an electrical circuit containing logic gates. This section examines the fundamental principles of quantum computation. These principles include quantum circuits and the basic building blocks for quantum circuits known as quantum gates. Quantum circuits are collections of quantum gates, for example the Hadamard gate, which are arranged so that they can perform some computational operation. The circuit model used to describe quantum algorithms is regarded as a universal language for describing more complicated quantum computations. The circuit model of computation uses universal quantum gates. Universal meaning that any quantum computation whatsoever can be expressed in terms of those gates. The language of quantum circuits provides an efficient and powerful language for describing quantum algorithms as it enables us to quantify the cost of an algorithm in terms of things like the total number of gates required and/or the circuit depth.

2.1 Qubit

A qubit is defined as the fundamental unit of information storage in quantum computation [1]. Though we typically treat qubits as mathematical objects to make generalization of theories a bit easier, it must be noted that qubits are actually physical objects. A qubit is a two state quantum-mechanical system, examples of qubit systems are the polarization states of a photon (H (horizontally polarized light) and V(vertically polarized light)) or the spin states of electrons (spin up and spin down) amongst others. In the atom model an electron can be found in either of two states, either the ground state or the excited state and by interacting with the system (light on the atom) we can move the electron in-between the states hence such a system can be thought of as being a qubit system.

In classical computation the fundamental unit for storing information is known as a bit, for example in an electronic system we can use current flow as a bit of information, *eg*, we can set the system to be in a "0" state if no current flows through the system and "1" if a current is detected. By doing this we can encode large amounts of information as a series of zero's and ones, this is how modern day computers encode information for processing. One can think of the qubit as being an analogous quantum unit for storing information which mathematically can be expressed as vectors in the Hilbert space, $|0\rangle$ or $|1\rangle$.

In a quantum mechanical system, for example the previous mentioned atom model, it is possible to have a state which is a combination of the two vectors which describe the system. This is known as a superposition and we say that the electron is in a superposition of the ground and excited state. Mathematically we express this state as $\alpha|0\rangle + \beta|1\rangle$ where α and β are defined as amplitudes, $\alpha, \beta \in \mathbb{C}$. The amplitudes are such that they satisfy the condition $|\alpha|^2 + |\beta|^2 = 1$. The superposition

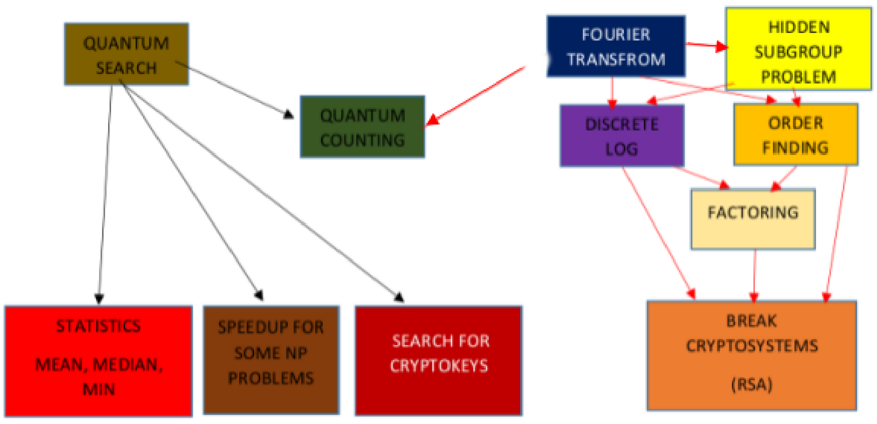


Figure 2.1: Branches of quantum search and quantum Fourier algorithms [1].

property is not present in classical computing and can be regarded as one of the main differences between bits and qubits.

2.2 Quantum algorithms

An algorithm can be defined as an ordered sequence of procedural operations for solving a problem within a finite number of steps. A quantum algorithm is a technique of problem solving which uses quantum resources for its implementation. These resources are quantum gates and quantum states. The algorithm is usually built using a quantum circuit model for easier reading. The field of quantum computation has become one of major interest in particular due to promises of a speed up compared to classical computers. These promises in speed-up become clearer when we consider two classes of quantum algorithms in particular, *i.e*, Shor's algorithm and Grover's algorithm. Shor's algorithm implements a quantum Fourier transform in solving the factoring and discrete logarithm problems providing an exponential speed up over the best known classical algorithms. Grover's search algorithm also provides a speed up in searching problem however only a quadratic speed up over known classical algorithms. This is still very impressive and gives much hope in the direction of quantum computation. Figure 2.1 extracted from Nielson and Chang's, "Quantum Computation and Quantum Information" textbook shows the importance of these two algorithms and how they extend to solving many other problems of interest [1]. The red arrows show extensions of the quantum Fourier transform algorithm and the black lines point towards extensions of the quantum search algorithms (Grover's algorithm in particular). The extensions are applications of the quantum search and quantum Fourier transform algorithm.

2.3 Complexity

Though there is some evidence that quantum computers are more computationally efficient compared to classical computers, the question as to whether quantum computers are conclusively more computationally efficient compared to classical computers is still an open one. This is because there are no known quantum and /or classical algorithms which solve NP complete problems efficiently. This should become clearer later. Computational complexity theory can be defined as the subject of classifying the difficulty of performing different computational tasks. It can better be described as the process of quantifying the amount of time memory expended by a computer when performing steps in an algorithm. Two variables are usually used to measure algorithmic complexity, *i.e*, time complexity **T** and space complexity **S** where, if the input has some size n bits then **T** and **S** are functions of n. The time complexity is measured in terms of $\mathbf{T}\ (n)$ which is the number of basic operations required to implement or execute an algorithm. The advantage of measuring the time complexity by counting the number of operations $\mathbf{T}\ (n)$ that it has to perform versus the actual execution time it has to take on the computer is that $\mathbf{T}\ (n)$ is system independent, meaning it is independent of the speed of the computer used to implement it hence the measure is fixed irrespective of whether you use a computer with greater processing capabilities or not. The space complexity is measured in terms of the number $\mathbf{S}\ (n)$ which is a measure of the amount of memory necessary to store the value of an integer.

For many algorithms it is difficult to quantify exactly the exact amount of memory or the exact number of operations required during execution as these are often dependant on input type, recursive nature of the algorithm and conditional structures such as "*if then*" and "*while*" statements. So typically computer scientists resort to evaluating what is referred to as the worst-case complexity of an algorithm. This means we look at the largest $\mathbf{T}\ (n)$ and $\mathbf{S}\ (n)$ which an algorithm can attain for any n. Exact upper bounds are usually abandoned in preference of asymptotic upper bounds (bounds describing worst case behaviour of the functions $\mathbf{T}\ (n)$ and $\mathbf{S}\ (n)$ as $n \to \infty$). This leads us to the "Big O" notation.

In computer science we write $f(n) = O(g(n))$ if there exists constants c such that $c \in \mathbb{R}^+$ and $n_0 \in \mathbb{N}$ such that

$$0 \leq f(n) \leq c.g(n)\ \forall\ n \geq n_o. \tag{2.1}$$

The function $g(n)$ is called the **asymptotic upper bound** for the function $f(n)$ as $n \to \infty$. A complexity class is a collection of computational problems with some common feature with respect to the computational resources needed to solve problems. The two important complexity classes are P and NP where P is the class of problems which can be solved in polynomial time (thus efficiently) on a classical computer whereas NP problems are a class of problems which can be checked quickly on a classical computer but there is no known efficient way to locate a solution in polynomial time or even polynomial resources. $P \subseteq NP$ since the ability to solve a problem implies the ability to check potential solutions however it is not clear that there are problems in NP which are not in P [1].

A traditional example used to distinguish between P and NP classes is the problem of finding prime factors for some integer n. There is no known efficient method of solving this problem on a classical computer and hence the problem is not in P however if one were told that some number p is a factor of n it would be very easy to verify hence the problem is in NP. It is known that $P \subseteq NP$, however it is not so clear as to whether $NP \subseteq P$ which creates a problem in determining if the two classes are different. It should be noted however that most researchers believe that NP contains problems which are not in P. This subclass of NP problems is known as **NP complete** (A problem is NP complete if it solution in a given time, t, allows us to solve all other NP problems in polynomial time) [1]. **P space** is a complexity class which consists of those problems which can be solved using resources which are few in spatial size but not necessarily in time.

Though **P space** is believed to be larger than NP and P, it has not yet been proven. NP class thus can be solved using randomized algorithms in polynomial time if a bound probability error is allowed to the solution. The theory of quantum computation poses interesting and significant challenges to the traditional notions of computation, however it is only until we have a functional quantum computer that we can make any conclusions.

2.4 Introduction to the quantum Fourier transform

At present finding the prime factorization of an n-bit integer using the best classical algorithm (number field sieve) is thought to require $\exp(\theta n^{\frac{1}{3}} \log(\frac{2}{3}n))$ operations. There are three values of θ relevant to different variations of the method [128], for the case when the algorithm is applied to numbers near a large power we take the value of $\theta \approx 1.52$, for the case where we apply the method to any odd positive number which is not a power $\theta \approx 1.92$ and for a version using many polynomials [129] $\theta \approx 1.90$. Because the number of steps required to compute such a problem

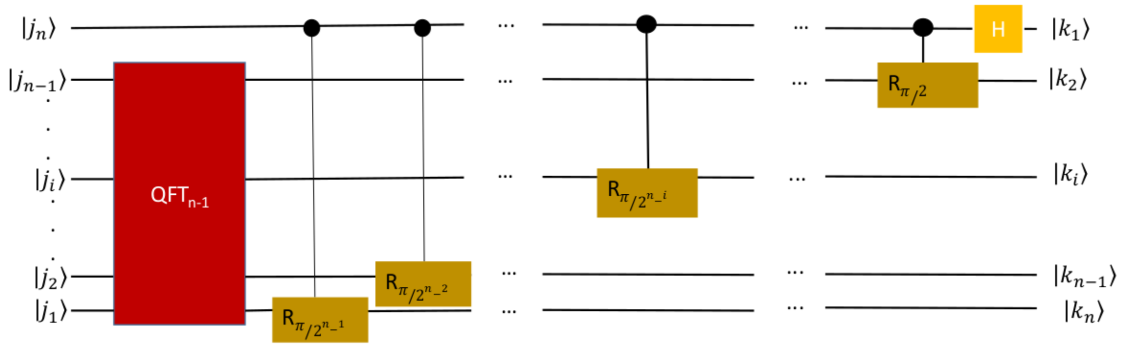


Figure 2.2: Quantum circuit for the implementation of a quantum Fourier transform. The picture however does not show the swap gates which are added at the end which reverse the order of the qubit[1]

.

increases exponentially with the size of the input the computation quickly becomes intractable on a classical computer, even for relatively small numbers. A similar problem computed on a quantum computer would require a number of operations of the order $O(n^2)\log n \log \log n$ [1]. Thus a quantum computer can factor an n-bit integer exponentially faster than the best known classical algorithms. This section looks at the quantum Fourier Transform which is a quantum algorithm for performing a Fourier transform of quantum mechanical amplitudes [1]. It is at the heart of many key algorithms such as factoring, order finding and phase estimation. It is worth noting that the quantum Fourier transform does not speed up the classical task of computing Fourier transforms of classical data but rather enables *phase estimation* which allows us to solve order-finding problem and the factoring problem [1].

2.4.1 Quantum Fourier transform

In mathematical notation, the Fourier transform (discrete in this case) takes a vector of complex numbers $a_0, a_2, \dots, a_{N-1}$, where N is the number of components of the vector and outputs another vector of complex numbers $b_0, b_1, \dots, b_{N-1}$, which is defined as below

$$b_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j e^{\frac{2\pi ijk}{N}}. \tag{2.2}$$

The quantum version of the Fourier transform essentially behaves in the same manner as the classical Fourier transform however it is different in that it operates on states. The quantum Fourier transform on an orthonormal basis $|0\rangle, |1\rangle, \dots, |N-1\rangle$ is defined to be a linear operator with the following action on the basis states [1]

$$|j\rangle \xrightarrow{\text{QFT}} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}} |k\rangle \tag{2.3}$$

The action of the quantum Fourier transform on an arbitrary state may be written as

$$\sum_{j=0}^{N-1} x_j |j\rangle \xrightarrow{\text{QFT}} \sum_{k=0}^{N-1} y_k |k\rangle \tag{2.4}$$

where the amplitudes y_k are the discrete Fourier transform of the amplitudes x_j. It should be noted that the Quantum Fourier Transform is a unitary operation.

Fig 2.2 shows a picture of the implementation of the quantum Fourier transform where R_θ are controlled phase gates in quantum computation. Though the Quantum Fourier transform is a

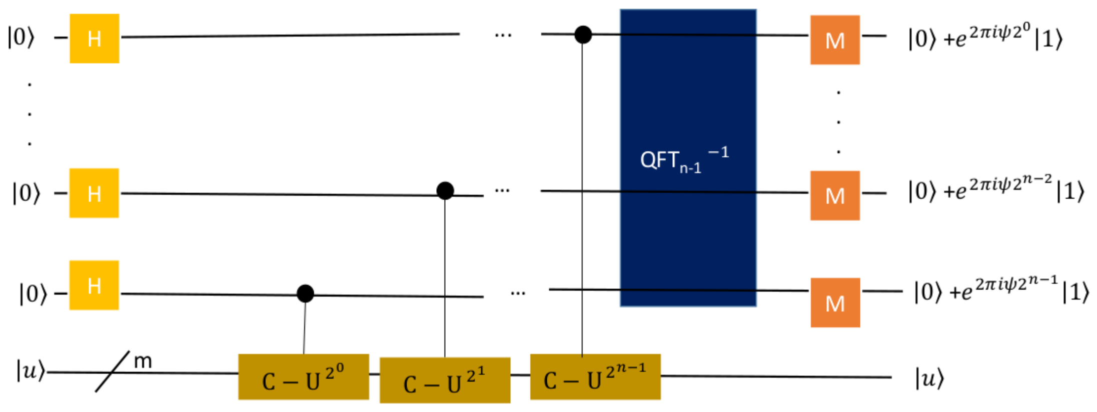

$|0\rangle$
H
$|0\rangle$
H
$|0\rangle$
H
$|u\rangle$
m
$C-U^{2^0}$
$C-U^{2^1}$
$C-U^{2^{n-1}}$
QFT_{n-1}^{-1}
M
M
M
$|0\rangle + e^{2\pi i\psi 2^0}|1\rangle$
$|0\rangle + e^{2\pi i\psi 2^{n-2}}|1\rangle$
$|0\rangle + e^{2\pi i\psi 2^{n-1}}|1\rangle$
$|u\rangle$

Figure 2.3: Quantum circuit for the implementation of a quantum Phase estimation algorithm[1]

pivotal tool, our main interest is in the algorithms within which it is used. These include the *phase estimation* algorithm and other algorithms which depend on the phase estimation algorithm.

2.4.2 Phase estimation algorithm

The algorithm uses a black box which performs a controlled-U^j operation (which applies the unitary operator U on the second register only if its corresponding control bit is $|1\rangle$) for integer j where $1 \leq j \leq 2^{t-1}$. Input qubits are initialized to zero and in order to successfully obtain ψ (the phase) accurate to n bits with probability of success at least $1-\epsilon$ we initialize $t = n + [\log(2 + \frac{1}{2\epsilon})]$ of these qubits. The algorithm requires that an eigenstate state $|u\rangle$ of the above unitary operator U^j to be prepared. The eigenvalue equation of the operator U and it's eigenstate $|u\rangle$ is given by

$$U|u\rangle = e^{2\pi i\psi}|u\rangle. \tag{2.5}$$

After $O(t^2)$ operations and a single call to the unitary operation U, the algorithm produces an n-bit approximation $\bar{\psi}$ to the phase ψ. The initial states are $|0\rangle^{\otimes t}|u\rangle^{\otimes m}$, where m is the number of auxiliary qubits. A Hadamard operation is then performed which produces a superposition of states represented as

$$\xrightarrow{\mathrm{H}^{\otimes t}\otimes \mathbb{1}^{\otimes m}} (\frac{1}{\sqrt{2}})^t (|0\rangle + |1\rangle)^{\otimes t}|u\rangle^{\otimes m} \tag{2.6}$$

For easier reading we can set $|u\rangle^{\otimes m} = |u\rangle$. The next step is to apply the above mentioned unitary controlled-U^{2^t} operator $\mathrm{C}-\mathrm{U}^{2^t}$ to the state in expression 2.6. We thus have

$$\mathrm{C}-\mathrm{U}^{2^{t-1}}(\frac{1}{\sqrt{2}})(|0\rangle + |1\rangle)|u\rangle \otimes \mathrm{C}-\mathrm{U}^{2^{t-2}}(\frac{1}{\sqrt{2}})(|0\rangle + |1\rangle)|u\rangle \otimes \ldots \otimes \mathrm{C}-\mathrm{U}^{2^0}(\frac{1}{\sqrt{2}})(|0\rangle + |1\rangle)|u\rangle. \tag{2.7}$$

This reduces to

$$(\frac{1}{\sqrt{2}})(|0\rangle|u\rangle + |1\rangle U^{2^{t-1}}|u\rangle) \otimes (\frac{1}{\sqrt{2}})(|0\rangle|u\rangle + |1\rangle U^{2^{t-2}}|u\rangle) \otimes \ldots \otimes (\frac{1}{\sqrt{2}})(|0\rangle|u\rangle + |1\rangle U^{2^0}|u\rangle). \tag{2.8}$$

From the eigenvalue equation (equation 2.5), the above equation becomes

$$(\frac{1}{\sqrt{2}})(|0\rangle|u\rangle + |1\rangle e^{2^{t-1}(2\pi i\psi_n)}|u\rangle) \otimes (\frac{1}{\sqrt{2}})(|0\rangle|u\rangle + |1\rangle e^{2^{t-2}(2\pi i\psi_n)}|u\rangle) \otimes \ldots \otimes (\frac{1}{\sqrt{2}})(|0\rangle|u\rangle + |1\rangle e^{2^0(2\pi i\psi_n)}|u\rangle). \tag{2.9}$$

After applying all the t controlled operations and adding kicking back phases to the control bits in the first register, equation 2.8 can be generalized to

$$(\frac{1}{\sqrt{2^t}})\sum_{k=0}^{2^t-1} e^{(2\pi i\psi)k}\left|k\right\rangle\left|u\right\rangle. \tag{2.10}$$

where $|k\rangle$ is a binary. The next step is to apply the quantum inverse Fourier transform on the first register from which we get

$$(\frac{1}{\sqrt{2^t}})\sum_{k=0}^{2^t-1}\sum_{l=0}^{2^t-1} e^{(2\pi i\psi)k}e^{2\pi ik\frac{l}{2^t}}\left|l\right\rangle\left|u\right\rangle. \tag{2.11}$$

Collecting the terms in the exponents together and performing a bit of algebraic manipulation we get

$$(\frac{1}{\sqrt{2^t}})\sum_{k=0}^{2^t-1}\sum_{l=0}^{2^t-1} e^{-2\pi ik\frac{l}{2^t}(l-2^t\psi)}\left|l\right\rangle\left|u\right\rangle. \tag{2.12}$$

We can approximate the phase ψ by rounding off the value $2^t\psi$ to the nearest integer, *i.e*, we can approximate $2^t\psi = \bar{\psi} + 2^t\delta$ where $\bar{\psi}$ is the nearest integer approximation to $2^t\psi$ and $2^t\delta$ is an error term defined such that $0 \leq \delta \leq \frac{1}{2^{t+1}}$. We can thus substitute the approximated terms in for the phase and we find the following

$$(\frac{1}{\sqrt{2^t}})\sum_{k=0}^{2^t-1}\sum_{l=0}^{2^t-1} e^{-2\pi ik\frac{l}{2^t}(l-\bar{\psi}-2^t\delta)}\left|l\right\rangle\left|u\right\rangle. \tag{2.13}$$

It is clear with a bit of algebraic manipulation that the above equation reduces to

$$(\frac{1}{\sqrt{2^t}})\sum_{k=0}^{2^t-1}\sum_{l=0}^{2^t-1} e^{-2\pi ik\frac{l}{2^t}(l-\bar{\psi})}e^{2\pi ik\frac{l}{2^t}\delta}\left|l\right\rangle\left|u\right\rangle. \tag{2.14}$$

The next step is to take a measurement on the first register in the basis $|\bar{\psi}\rangle$

$$(|\bar{\psi}\rangle\langle\bar{\psi}|)(\frac{1}{\sqrt{2}})^t\sum_{k=0}^{2^t-1}\sum_{l=0}^{2^t-1} e^{-2\pi ik\frac{l}{2^t}(l-\bar{\psi})}e^{2\pi ik\frac{l}{2^t}\delta}\left|l\right\rangle\left|u\right\rangle \tag{2.15}$$

which gives us the state $|\bar{\psi}\rangle \otimes |u\rangle$. Measurement in the computational basis on the first register yields the desired result with probability

$$\Pr(|\bar{\psi}\rangle) = |\langle\bar{\psi}|(\frac{1}{\sqrt{2}})^t\sum_{k=0}^{2^t-1}\sum_{l=0}^{2^t-1} e^{-2\pi ik\frac{l}{2^t}(l-\bar{\psi})}e^{2\pi ik\frac{l}{2^t}\delta}\left|l\right\rangle|^2, \tag{2.16}$$

This reduces to

$$\Pr(|\bar{\psi}\rangle) = \frac{1}{2^{2t}}|e^{2\pi ik\frac{l}{2^t}\delta}|^2, \tag{2.17}$$

$$\Pr(|\bar{\psi}\rangle) = \begin{cases} 1, & \text{for } \delta = 0, \\ \frac{1}{2^{2t}}|\frac{1-e^{2\pi i2^n\delta}}{1-e^{2\pi i\delta}}|, & \text{for } \delta \neq 0. \end{cases}$$

When the error δ is zero, we will get a very good approximation with a probability of 1 and even if the error is not zero, then we will get a good approximation with a relatively high probability. In depth analysis of the complexity of the phase estimation algorithm can be found in Nielsen and Chang's book [1]. Fig 2.3 shows the quantum circuit for the phase estimation algorithm.

2.4.3 Applications of the phase estimation algorithm and QFT: Order finding and factoring

The fast quantum algorithms for order-finding and factoring are interesting because they provide evidence for the idea that quantum computers may be inherently more powerful than classical computers. The order-finding and factoring algorithms thus provide a credible challenge to the strong Church-Turing thesis, which states that any model of computation can be translated into an equivalent computation involving a probabilistic Turing machine with at most a polynomial increase in the number of elementary operations required [1]. Efficient algorithms for order-finding and factoring can be used to break the RSA public-key cryptosystem which justifies interest in the algorithms. It is also true that both problems are of sufficient intrinsic worth to justify interest in any novel algorithm, be it classical or quantum.

2.4.4 Order-finding

For some positive integers x and N, where $x < N$ and where x and N have no common factors, we define the order of x modulo N to be the least positive integer, r, such that $x^r = 1(\mathrm{mod} N)$ and the order-finding problem is the one to determine the order for some specified x and N.

The algorithm takes uses a black box which performs a controlled-$U_{x,N}$ operation for x co-prime (two numbers m and p are said to be co-prime if the only positive integer (factor) that divides both of them is 1) to the L-bit number N. The unitary $U_{x,N}$ is given by the operation

$$|j\rangle\,|k\rangle \to |j\rangle\,\left|x^j k \bmod N\right\rangle. \tag{2.18}$$

We initialize $t = n + [\log(2 + \frac{1}{2\epsilon})]$ qubits to $|0\rangle$ and L qubits to $|1\rangle$. After $O(L^3)$ operations the algorithm outputs the least integer $r > 0$ such that $x^r = 1(\mathrm{mod} N)$ with a probability of order of magnitude, $O(1)$. The initial states are $|0\rangle^{\otimes t}|1\rangle^{\otimes L}$. A Hadamard operation is performed on the first register which produces a superposition of states represented as

$$\xrightarrow{\mathrm{H}^{\otimes t}\otimes \mathbb{1}^{\otimes L}} (\frac{1}{\sqrt{2}})^{t}\,(|0\rangle + |1\rangle)^{\otimes t}|1\rangle^{\otimes L}. \tag{2.19}$$

For easier reading we set $|1\rangle^{\otimes L} = |1\rangle$. The next step is to apply the above mentioned unitary controlled-$U_{x,N}^{2^t}$ operator. We thus have

$$\mathrm{C}-\mathrm{U}_{x,N}^{2^{t-1}}(\frac{1}{\sqrt{2}})(|0\rangle + |1\rangle)|1\rangle \otimes \mathrm{C}-\mathrm{U}_{x,N}^{2^{t-2}}(\frac{1}{\sqrt{2}})(|0\rangle + |1\rangle)|1\rangle \otimes \ldots \otimes \mathrm{C}-\mathrm{U}_{x,N}^{2^{0}}(\frac{1}{\sqrt{2}})(|0\rangle + |1\rangle)|1\rangle. \tag{2.20}$$

This reduces to

$$(\frac{1}{\sqrt{2}})(|0\rangle\,|1\rangle + |1\rangle\,U_{x,N}^{2^{t-1}}|1\rangle) \otimes (\frac{1}{\sqrt{2}})(|0\rangle\,|1\rangle + |1\rangle\,U_{x,N}^{2^{t-2}}|1\rangle) \otimes \ldots \otimes (\frac{1}{\sqrt{2}})(|0\rangle\,|1\rangle + |1\rangle\,U_{x,N}^{2^{0}}|1\rangle), \tag{2.21}$$

it follows from equation 2.18 that

$$(\frac{1}{\sqrt{2}})(|0\rangle\,|1\rangle + |1\rangle\left|x^{2^{t-1}}\mathrm{mod}N\right\rangle) \otimes (\frac{1}{\sqrt{2}})(|0\rangle\,|1\rangle + |1\rangle\left|x^{2^{t-2}}\mathrm{mod}N\right\rangle) \otimes \ldots \otimes (\frac{1}{\sqrt{2}})(|0\rangle\,|1\rangle + |1\rangle\,|(x^0\mathrm{mod})N\rangle. \tag{2.22}$$

This above equation reduces to

$$\frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle\left|x^j \mathrm{mod}N\right\rangle. \tag{2.23}$$

It can be shown that the eigenstates of $U_{x,N}$ are $|u_s\rangle$ which is defined as

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{2\pi \frac{s}{r} k} \left| x^k \text{mod} N \right\rangle . \tag{2.24}$$

$U_{x,N} |u_s\rangle = e^{2\pi i s \frac{j}{r}} |u_s\rangle$ and $|u_s\rangle$ is an eigenstate of the unitary $U_{x,N}$. We thus substitute in the eigenstate $|u_s\rangle$ into equation 2.21 and get

$$\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle \left| x^j \text{mod} N \right\rangle = \frac{1}{\sqrt{r 2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j / r} |j\rangle |u_s\rangle . \tag{2.25}$$

The ratio $\frac{s}{r}$ is the equivalent of the phase in the phase estimation algorithm. Applying the inverse quantum Fourier transform on the first register we get

$$(\frac{1}{\sqrt{2^t}}) \sum_{k=0}^{2^t-1} \sum_{l=0}^{2^t-1} e^{(2\pi i \frac{s}{r})k} e^{2\pi i k \frac{l}{2^t}} |l\rangle |u_s\rangle . \tag{2.26}$$

Collecting the terms in the exponents together and performing a bit of algebraic manipulation we get

$$(\frac{1}{\sqrt{2^t}}) \sum_{k=0}^{2^t-1} \sum_{l=0}^{2^t-1} e^{-2\pi i k \frac{l}{2^t} (l - 2^t \frac{s}{r})} |l\rangle |u_s\rangle . \tag{2.27}$$

We can approximate the phase s/r by rounding off the value $2^t s/r$ to the nearest integer, *i.e*, we can approximate $2^t s/r = \bar{s/r} + 2^t \delta$ where $\bar{s/r}$ is the nearest integer approximation to $2^t \psi$ and $2^t \delta$ is an error term defined such that $0 \leq \delta \leq \frac{1}{2^{t+1}}$. The approximations thus give us

$$(\frac{1}{\sqrt{2^t}}) \sum_{k=0}^{2^t-1} \sum_{l=0}^{2^t-1} e^{-2\pi i k \frac{l}{2^t} (l - \frac{\bar{s}}{r} - 2^t \delta)} |l\rangle |u_s\rangle . \tag{2.28}$$

It is clear that the above equation reduces to

$$(\frac{1}{\sqrt{2^t}}) \sum_{k=0}^{2^t-1} \sum_{l=0}^{2^t-1} e^{-2\pi i k \frac{l}{2^t} (l - \frac{\bar{s}}{r})} e^{2\pi i k \frac{l}{2^t} \delta} |l\rangle |u_s\rangle \tag{2.29}$$

The next step is to take a measurement on the first register in the basis $\left| \frac{\bar{s}}{r} \right\rangle$,

$$(\left| \frac{\bar{s}}{r} \right\rangle \left\langle \frac{\bar{s}}{r} \right|)(\frac{1}{\sqrt{2}})^t \sum_{k=0}^{2^t-1} \sum_{l=0}^{2^t-1} e^{-2\pi i k \frac{l}{2^t} (l - \frac{\bar{s}}{r})} e^{2\pi i k \frac{l}{2^t} \delta} |l\rangle |u_s\rangle , \tag{2.30}$$

which gives us the state $\left| \frac{\bar{s}}{r} \right\rangle \otimes |u\rangle$. Measurement in the computational basis on the first register yields the desired result with probability

$$\Pr(\left| \frac{\bar{s}}{r} \right\rangle) = |\left\langle \frac{\bar{s}}{r} \right| (\frac{1}{\sqrt{2}})^t \sum_{k=0}^{2^t-1} \sum_{l=0}^{2^t-1} e^{-2\pi i k \frac{l}{2^t} (l - \frac{\bar{s}}{r})} e^{2\pi i k \frac{l}{2^t} \delta} |l\rangle |^2 \tag{2.31}$$

This reduces to

$$\Pr(\left| \frac{\bar{s}}{r} \right\rangle) = \frac{1}{2^{2t}} |e^{2\pi i k \frac{l}{2^t} \delta}|^2 \tag{2.32}$$

$$\Pr(\left| \frac{\bar{s}}{r} \right\rangle) = \begin{cases} 1, & \text{for } \delta = 0, \\ \frac{1}{2^{2t}} |\frac{1 - e^{2\pi i 2^n \delta}}{1 - e^{2\pi i \delta}}|, & \text{for } \delta \neq 0. \end{cases}$$

This means that when the error δ is zero we will get a very good approximation with a probability of 1 and even if the error is not zero then we will get a good approximation with a relatively high probability. Indepth analysis of the complexity of the phase estimation algorithm can be found in Nielsen and Chang's book [1]. Another prominent application of the quantum Fourier transform is factoring. The factoring problem can be reduced to the order-finding problem and hence relies on the algorithm above. The factoring algorithm otherwise known as Shor's algorithm will not be discussed in this thesis despite it's great value [1].

2.4.5 Period-finding

Problem description: let the function f be a periodic function which produces a single bit as output and is defined such that $f(x+r) = f(x)$ for some unknown $0 < r < 2L$, where $x, r \in 0, 1, 2, \ldots$. The problem is to find r. Period finding is used in Shor's algorithm.

The algorithm uses a black box which performs the operation $U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. It also needs an extra register initialized to $|0\rangle$ which stores the functional value. We initialize $t = O(L + \log(\frac{1}{\epsilon}))$ qubits to $|0\rangle$. After $O(L^2)$ operations and a single application of the unitary U the algorithm outputs the least integer $r > 0$ such that $f(x+r) = f(x)$ with a probability of $O(1)$ [1].

The initial states is $|0\rangle^{\otimes t}|0\rangle$. The procedure is identical to that of implementing the phase estimation and order finding algorithms discussed in the preceding sections and hence details will be left out. A Hadamard operation is performed on the initial state which produces a superposition of states represented as

$$\xrightarrow{\mathrm{H}^{\otimes \mathrm{t}} \otimes \mathbb{1}} \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|0\rangle . \tag{2.33}$$

The next step is to apply the above mentioned unitary operator U on the state in equation 2.31 which results in the following

$$\xrightarrow{\mathbb{1}^{\otimes t} \otimes \mathrm{U}} \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|f(x)\rangle . \tag{2.34}$$

It can be shown that the eigenstate of the unitary U is the state $\left|\hat{f(l)}\right\rangle$ which is defined such that it satisfies $|f(x)\rangle = \frac{1}{\sqrt{r}} \sum_{t=0}^{r-1} e^{2\pi i l \frac{x}{r}} \left|\hat{f(l)}\right\rangle$. From this it follows that

$$\frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|f(x)\rangle = \frac{1}{\sqrt{r2^t}} \sum_{l=0}^{r-1} \sum_{x=0}^{2^t-1} e^{2\pi i l x/r} |x\rangle \left|\hat{f(l)}\right\rangle . \tag{2.35}$$

Applying the inverse quantum Fourier transform on the first register, the exact same procedure is followed as from the phase estimation and order finding algorithms where we approximate the value of the ratio $\frac{l}{r}$. Once we have a value $\frac{l}{r}$ we can apply the continued fractions algorithm to get the value of r. The period finding algorithm is based on phase estimation and is nearly identical to the algorithm for quantum order-finding with the exception of the introduction of the $\left|\hat{f(l)}\right\rangle = \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} e^{2\pi i l x/r} |f(x)\rangle$ which is just a Fourier transform.

2.4.6 Discrete logarithms

Problem description: Let function $f(x_1, x_2)$ be a periodic function which produces a single bit as output and is defined such that $f(x_1, x_2) = a^{sx_1+x_2}\mathrm{modN}$ where all the variables are integers. r is taken to be the smallest positive integer such that $a^r \mathrm{mod} N = 1$. $f(x_1 + l, x_2 - ls) = f(x_1, x_2)$ and hence the period is a 2-tuple, $(l - ls)$, for integer l. The problem is to find r. The algorithm uses a black box (Unitary) which performs the operation

$$U|x_1\rangle|x_2\rangle|y\rangle = |x_1\rangle|x_2\rangle|y + f(x_1, x_2)\mathrm{mod}\ 2\rangle \tag{2.36}$$

For the function $f(x_1, x_2) = b^{x_1}a^{x_2}$ we need an extra register initialized to $|0\rangle$ which stores the functional value. We initialize $t = O(\log r + \log(\frac{1}{\epsilon}))$ qubit registers to $|0\rangle$. After $O([\log r]^2))$ operations and a single application of the unitary U the algorithm outputs the least integer s such that $a^s = b$ with a probability of $O(1)$. Applying the algorithm known as the generalized continued fractions algorithm [1], the value of r can be calculated. The initial states look as follows $|0\rangle |0\rangle |0\rangle$. The algorithm uses the exact same ideas as the previous algorithms and hence will not be discussed to any detail. It is included merely to demonstrate the capacity of the quantum Fourier transform.

2.5 Grover search algorithm

Given an oracle with N allowed inputs which can be for example cities on a map we may wish to find a certain feature or one city with a certain property. We can make it such that the target city is marked with a "1" whereas all other cities are marked with a "0". Thus given a list of N inputs the task is to find the input with the required feature. With a classical algorithm this requires $O(N)$ queries. The quantum search algorithm of Lov Grover otherwise known as Grover's algorithm achieves this task using $O(\sqrt{N})$ queries which is substantially less than the classical algorithms [3]. Grover's algorithm has been generalized to search in the presence of multiple solutions [12]. Grover's algorithm has many applications including finding the global minimum of an arbitrary function [13, 14, 15], evaluating the sum of an arbitrary function [12, 16, 17], finding approximate definite integrals [18] and converge to a fixed-point [20, 21]. This has algorithm has subsequently been generalized to take advantage of alternative initial states [22], *i.e*, nonuniform probabilistic states [23]. Grovers algorithm has aswell been used in amplitude estimation [24], which forms the core of most known quantum algorithms related to collision finding and graph properties. Other applications of Grover's algorithm include the speeding up the solution to NP-complete problems such as 3-SAT [25], speedup of other constraint satisfaction problems [26, 27, 28] amongst others. Spatial search is a problem closely related to the Grover's algorithm but is harder. In the spatial search problem the database queries are limited by some graph structure. It has been shown though that for sufficiently well-connected graphs, $O(\sqrt{N}))$ quantum query complexity is possible [29, 30].

2.5.1 Grover search algorithm description

The algorithm uses a black box oracle O which performs the transformation $O|x\rangle |q\rangle = |x\rangle |q \oplus x\rangle$, where $f(x) = 0$ for all $0 \leq x < 2n$ except x_0 for which $f(x_0) = 1$. After an $O(\sqrt{N})$ operations the algorithm succeeds with probability of order of magnitude, $O(1)$. We initialize $n + 1$ qubits in the state $|0\rangle$.

The initial states look as follows $|0\rangle^{\otimes n} |0\rangle$. A $H^{\otimes n}$ operation is performed on the first n qubits which produces a superposition of states and HX is applied to the second qubit as shown

$$\xrightarrow{\mathrm{H^{\otimes n} \otimes HX}} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \, [\frac{|0\rangle - |1\rangle}{\sqrt{2}}]. \tag{2.37}$$

The next step is to apply the apply the Grover iteration $\frac{\pi\sqrt{2^n}}{4}$ times, details are in Nielson and Chang's textbook [1]. For easier reading we set $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle = |\mathbf{x}\rangle$. Hence the above equation becomes $|\mathbf{x}\rangle |0\rangle - |\mathbf{x}\rangle |1\rangle$. We begin by applying the oracle, O

$$\rightarrow O\frac{1}{\sqrt{2}} |\mathbf{x}\rangle |0\rangle - O\frac{1}{\sqrt{2}} |\mathbf{x}\rangle |1\rangle . \tag{2.38}$$

By the definition of the oracle it follows

$$\rightarrow \frac{1}{\sqrt{2}} |\mathbf{x}\rangle |f(\mathbf{x})\rangle - \frac{1}{\sqrt{2}} |\mathbf{x}\rangle |1 \oplus f(\mathbf{x})\rangle , \tag{2.39}$$

$$= \begin{cases} \frac{1}{\sqrt{2}}(|\mathbf{x}\rangle |1\rangle + |\mathbf{x}\rangle |0\rangle) = -|\mathbf{x}\rangle \otimes |-\rangle , & \text{if } f(x) = 1, \\ \frac{1}{\sqrt{2}}(|\mathbf{x}\rangle |0\rangle + |\mathbf{x}\rangle |1\rangle) = |\mathbf{x}\rangle \otimes |-\rangle , & \text{if } f(x) = 0. \end{cases}$$

We can relabel the state for the case where $f(x) = 1$ as $-|\mathbf{t}\rangle \otimes |-\rangle$, this is called the target state and the relabelling is meant for clarity. We will ignore the $|-\rangle$ since it does not change during the implementation of the Grover algorithm. The next step is to apply the diffusion operator to the first register. Setting $\mathrm{H}^{\otimes \mathrm{n}}|0\rangle^{\otimes n} = |\phi\rangle$ it can be shown that the output state after the implementation of the oracle can be written as $|\phi\rangle - \frac{2}{\sqrt{N}}|\mathbf{t}\rangle$. We then apply the diffusion operator on this state, some trivial conditions are added to make the understanding of the calculation easier, *i.e*,

$$\langle t|\phi\rangle = \langle\phi|t\rangle = \frac{1}{\sqrt{N}}, \tag{2.40}$$

$$\langle\phi|\phi\rangle = 1. \tag{2.41}$$

The application of the diffusion operator is shown below

$$((2|\phi\rangle\langle\phi| - 1) \otimes \mathbb{1}))(|\phi\rangle - \frac{2}{\sqrt{N}}|\mathbf{t}\rangle) = \frac{N-4}{N}|\phi\rangle + \frac{2}{\sqrt{N}}|\mathbf{t}\rangle. \tag{2.42}$$

We can see that the amplitude of measuring the target state has gone up from $\frac{1}{\sqrt{N}}$ to after applying the oracle, O and the diffusion operator which can be labelled, D. It can be shown that after a number of operations $O(\sqrt{N})$, the algorithm succeeds to find the target state with probability of order of magnitude, $O(1)$. Finally we measure in the computational basis to get the target state with a high probability. Indepth analysis of the complexity of the Grover algorithm and it's applications can be found in Nielsen and Chang's book [1].

2.5.2 Geometric explanation

If we consider a 2 dimensional plane through the 2^n dimensional Hilbert space of the n qubits, we can define a plane containing the target state $|\mathbf{t}\rangle$ and one containing the superposition $\sum_{\phi_1,\ldots,\phi_n=0}^{1}|\phi\rangle$, these are nearly orthogonal but never quite so as their scalar product is $\frac{1}{\sqrt{N}}$. We then define a state which is orthogonal to $|\mathbf{t}\rangle$ and call it $|\alpha\rangle$. Let us then consider a family of states which lie in this plane which can be expressed as $\cos\theta|\mathbf{t}\rangle + \sin\theta|\alpha\rangle$ (θ is a real parameter) and hence the state we are looking for, $|\mathbf{t}\rangle$, is within this parametrised plane. When we perform the first subroutine namely the Hadamard operation we create a superposition state $\sum_{\psi_1,\ldots,\psi_n=0}^{1}|\psi\rangle$, where $\theta = \sin^{-1}(\frac{1}{\sqrt{N}})$. For a general state of the form $\cos\theta|\mathbf{t}\rangle + \sin\theta|\alpha\rangle$ the effect of a Grover iteration can be summarised into two subroutines, the first being the marking subroutine which changes the sign of the coefficient of the target state $|\beta\rangle$, the coefficient in this case being $\sin\theta$ whilst all other computation basis states are not changed. This operation geometrically is the equivalent of reflecting the state about the horizontal axis which in this case will be the axis with the $|\alpha\rangle$ state. After the marking subroutine we have the diffusion operation which has the effect of reflecting the state about the superposition state $\sum_{\phi_1,\ldots,\psi_n=0}^{1}|\phi\rangle$. The product of these two reflections can be shown to be a rotation through an angle $2\sin^{-1}(\frac{1}{\sqrt{N}})$ in the anticlockwise direction, that is towards $|\mathbf{t}\rangle$. Each iteration takes us a step closer towards $|\mathbf{t}\rangle$ until we go past it and hence it is of essence to choose the right number of iterations. A geometric picture of the the functioning of the Grover algorithm can be seen in figure 2.4 .

2.5.3 Grover summary

Suppose you have a database (for example a phonebook) where the entries of this database have been randomly captured and hence it is unstructured and we wish to search for a particular entry in this database then on a classical computer one would need $O(\mathrm{N})$operations to find an entry in the database. The solution in this case is easy to recognise however very difficult to find, a common feature of problems in the NP computational class (typically these problems are resolved best through an exhaustive search as typically no more efficient classical algorithm can be found)[1].

There exists a quantum algorithm which can solve this particular problem with a quadratic speed up and this algorithm is known as the Grover Search Algorithm which only requires $O(\sqrt{N})$

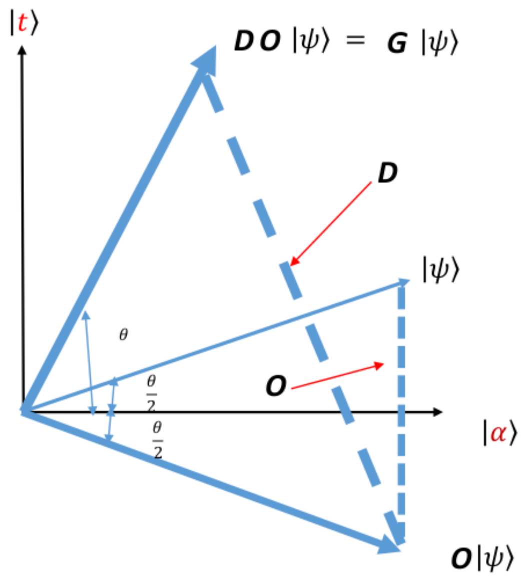


Figure 2.4: The Grover operator, G, rotates the superposition state $|\psi\rangle$ by an angle θ towards the target state $|\mathbf{t}\rangle$. The Grover operator is composed of two operators, *i.e*, the oracle operation O which reflects the state about the state $|\alpha\rangle$ (which is a state orthogonal to the target state) and a Diffusion operation, D, which rotates about the superposition. After repeated Grover iterations, the state vector gets close to the target state $|\mathbf{t}\rangle$, in this case [1]

operations. To get a numerical idea of the efficiency of the Grover algorithm over its classical counterparts, consider that if one had a budget of a million operations, a classical algorithm is limited to searching through about a million (unsorted, unstructured) possibilities whereas the Grover search algorithm can use those million operations to search through a trillion possibilities [46]. The Grover search algorithm has many applications outside of unstructured database searches including quantum counting and speed up in solution of other NP problems like the travelling salesman problem[45]. Beginning the description for the Grover algorithm we say, for $N = 2^n$ we are given an arbitrary

$$x_j \in \{0,1\}^{\otimes n} \tag{2.43}$$

for which we want to find a $j \in 1, 2, ..., N$ such that $x_j = 1$ and to return “nothing” or “no solution” if x_j is anything else.

It must be noted however that Grover’s algorithm doesn’t search through lists but rather it searches through function inputs, *i.e*, Grover’s algorithm takes a function, searches through the implicit list of possible inputs to that function, and returns the single input that causes the function to return true with high probability. There are different cases in which we can consider one true input and one satisfying more than one true input, in which case we need a variant of Grover’s algorithm [1].

Chapter 3

Classical optimization methods

A large number of problems can be referred to as optimization problems and this chapter aims at classifying these. This chapter attempts to give a very brief scope of the field of mathematical optimization. Instead of a comprehensive description of this vast research field this chapter merely aims to give a light flavour of the kind of topics one may encounter when studying classical optimization methods.

3.1 Applications of mathematical optimization

The branch of mathematics and computer science known as optimization has an abundance of practical applications. For this reason amongst others there has been and continues to be a considerable amount of research in this area. In electrical engineering, mathematical optimization techniques are used in space mapping design of microwave structures,[53], active filter design[54], electromagnetics-based design, stray field reduction in superconducting magnetic energy storage systems and handset antennas [55]. In solving rigid body mechanics problems physicists and engineers resort to mathematical programming techniques founded in mathematical optimization.

Mathematical optimization techniques as well surface in microeconomics when solving problems such as the utility maximization problem and its dual problem, the expenditure minimization problem. In macroeconomics dynamic stochastic general equilibrium (dsge) models that describe the dynamics of the whole economy and which which rely heavily on mathematical optimization are frequently used [106]. Other popular fields which rely on mathematical optimization are operations research, civil engineering, agriculture, geophysics, molecular modelling and control engineering.

3.2 Branches of mathematical optimization

The topic of mathematical optimization is a very large and well researched one. The primary reason being that it is a very practical and very applied topic. It surfaces in about every subject known primarily in mathematics, physics, computer science and engineering amongst many other fields. This is because each of these fields at some point or the other require a selection of the best element with regard to some conditions from a set of alternatives. This selection usually is of a minimum or maximum of a function.

There are different types of problems with different conditions which need to be solved. This results in many sub fields of the topic. This chapter will list as many of the major sub fields as possible as it is important that we classify which problems exist and which methods can be used to rectify them. The first sub field of mathematical optimization is convex programming which studies the problem of minimizing convex functions over convex sets.

Convex programming can be viewed as a case of nonlinear programming or as a generalization of linear programming or convex quadratic programming [32]. A convex minimization problem is thus written as

$$\text{minimize } f(x) \tag{3.1}$$

$$\text{subject to } g_i(x) \leq 0$$

where the functions $g_i(x)$ are convex

$$\text{and } h_i(x) = 0$$

where the functions $h_i(x)$ are affine. A detailed outline of convex functions is given in the Appendix A.

Conic programming: A sub-field of convex optimization which studies a class of structured convex optimization problems called conic optimization problems. These problems involve minimizing a convex function over the intersection of an affine subspace and a convex one [37]. Geometric programming: A method used to achieve the best outcome (for example maximum profit) in a mathematical model in which the objective and inequality constraints expressed as polynomial and equality constraints as monomials can be transformed into a convex problem. Geometric programs are in general not convex optimization problems but can be transformed into such [36].

The problem can be expressed mathematically as

$$\text{minimize } f^T x \tag{3.2}$$

subject to

$$||A_i x + b_i||_2 \leq {c_i}^T + d_i, i = 1, \ldots, m$$

$$Fx = g$$

where the problem parameters are $\mathbf{f} \in \mathbb{R}^{\mathbf{n}}$, $A_i \in \mathbb{R}^{n_i \times n}$, $b_i \in \mathbb{R}^{n_i}$, $c_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, $F \in \mathbb{R}^{p \times n}$ and $g \in \mathbb{R}^p$. x is the optimization variable.

Linear programming: It is a type of convex programming which studies the specific cases where the objective function is linear and the constraints specified using only linear equalities and inequalities [40]. The sets are polyhedrons or polytopes they are bounded.

Linear programs are problems that can be expressed in canonical form as

$$\text{maximize } c^T x, \tag{3.3}$$

$$\text{subject to } Ax \leq b,$$

$$\text{and } x \geq 0,$$

where x represents the vector of variables, c and b are vectors of coefficients, A is a matrix of coefficients.

Second order cone programming: A type of programming which studies non-linear convex problems which also include linear and (convex) quadratic programs as special cases but are less general than semi-definite programs where a linear function is minimized over the section of an affine set and the product of the second order (quadratic) cones [56].

Semi-definite programming: A branch of convex optimization where a linear objective function defined over the intersection of the cone of positive semi -definite matrices with affine space is optimized [33].

There is a multitude of optimization problems which are not necessarily convex problems. The list of problems is summarized here. Integer Programming: A mathematical optimization problem in which decision variables are integers, it is called a mixed integer program if some of the variables are allowed to be fraction and a pure integer program if all of the values are allowed. They are notoriously difficult to solve as there is in general no efficient algorithm for their solution [57].

Quadratic programming: A mathematical method for optimizing a quadratic objective function of several variables subject to bounds, linear equality and inequality constraints [58]. It is a subclass of non-linear programming.

Fractional programming: A generalization of linear fractional programming in which the objective function is a ratio of two functions which are generally non-linear. Recent developments have been made to this method [59].

Non-linear programming: A method of solving optimization problems with equality and inequality constraints and in which the objective function is non linear [38]. The problem can be convex or non-convex.

Stochastic programming: A framework for modelling optimization problems which involve uncertainty in the coefficients (parameters). This means that for instance in deterministic programming the coefficients are known numbers whereas in stochastic programming these coefficients are unknown and instead have a probabilistic distribution present [39].

Robust programming: Methodology for handling optimization problems with uncertain data which is represented with deterministic variability in the values of the parameters of the problem or it's solution [61].

Combinatorial optimization: A framework for finding an optimal object from a finite set of objects, it works in the domain of discrete feasible solutions or can be reduced to discrete. The problems involved typically can not be solved by exhaustive methods for example the travelling salesman problem and or the minimum spanning tree problem [66].

Stochastic optimization: A collection of methods for minimizing or maximizing an objective function in the presence of randomness [60].

Infinite-dimensional optimization: Optimization problem in which the optimal solution is not a number or a vector but rather a function or some other continuous structure [118].

Heuristics and meta heuristics: A technique in mathematics and computer science for solving a search optimization problem when classical methods are too slow or cannot detect a solution [35].

Constraint satisfaction: A technique of finding an optimal solution to a set of constraints which impose the conditions which the variables of the problem should satisfy and the bijective function f is a constant [63].

Space mapping: A technique intended for optimization of engineering models which involve expensive function evaluations, it is used to model and optimize an engineering system to high fidelity by exploiting a surrogate model [62].

The techniques of the following subfields are designed primarily for decision making over time meaning they perform optimization in dynamic contexts.

Calculus of variations: Branch of mathematics which deals with optimizing functionals [34].

Optimal control: Is a technique of mathematical optimization for deriving control policies, it is an extension of calculus of variations [64].

Dynamic programming: A method for solving complex optimization problems by breaking them down into simpler sub-problems [65].

Mathematical programming with equilibrium constraints: Study of constrained optimization problems in which the constraints include variational inequalities or complementaries [67].

3.3 Computational optimization techniques

We have looked at the classes of optimization problems so in this section we will look at the algorithms which are used to solve these problems. The algorithms used in computation are typically divided into two major categories, *i.e*, iterative algorithms and heuristic algorithms. Iterative methods are mathematically derived methods of search whereas heuristics are described as methods of learning which employ a practical method, for example the ant colony optimization technique.

The use of iterative algorithms is quite common in solving optimization problems. Iterative methods converge to a solution after a finite number of steps. Convergence of iterative methods is not always guaranteed, so we use heuristics to provide approximate solutions to some of these problems. The best algorithm to solve a problem usually depends on the function itself.

Iterative algorithms can be categorised into those which evaluate or approximate Hessians for example Newtons method [41], sequential quadratic programming [120] and interior point methods [121]. Those which evaluate or approximate gradients for example gradient descent [122], ellipsoid method [123] and simultaneous perturbation stochastic approximation [124]. The other group of iterative methods includes methods that evaluate function values which include interpolation methods and patterns search methods.

Apart from the iterative methods we also have heuristic methods. Heuristics are defined as techniques designed for solving a problem more quickly when classic methods are too slow and or for finding an approximate solution when classic methods fail to find any exact solution. These is a large number of such algorithm, *e.g*, particle swarm optimization algorithm [125], gravitational search algorithm [126] and artificial bee colony algorithm [131].

Chapter 4

Quantum optimization algorithms

This section summarises currently known quantum algorithms which are used in quantum optimization. It lists algorithms which are used in different branches of quantum optimization. Due to the difficulty of constructing quantum algorithms there aren't as many available for solving all our optimization problems.

4.1 Quantum optimization algorithms

There are many quantum algorithms which offer speedups ranging from polynomial through superpolynomial to exponential. These algorithms can be classified as being oracular, simulation (approximation) based algorithms or algebraic (number theoretic)algorithms.

Amongst oracular algorithms there exist searching (optimizing) algorithms which offer a polynomial speed-up. Examples of these algorithms are included in the references [91, 88, 89], in which single query quantum algorithms for finding the minima of basins based on Hamming distance are given. Another example of an oracle based quantum algorithm is Stephen Jordan's quantum numerical gradient search algorithm which can extract all d^2 matrix elements of the quadratic form using $O(d)$ queries [85]. The same algorithm presented in [85] can extract all d^n nth derivatives of a smooth function of d variables in $O(d^{n-1})$ queries which is a considerable leap from the classical algorithm. In the papers [90] and [91] we find quantum algorithms which extract quadratic forms and multilinear polynomials in d variables over a finite field. This is achieved with factor of d fewer quantum queries than are required classically. We include a summary of optimization algorithms which we list in the following section below.

4.2 Summary of quantum optimization algorithms

Algorithm 1: Ordered Search

Problem: Given an oracle which has access to a list of N sorted numbers in an order from least value to greatest value you may want to find out where in the list a number x would fit.

Classically the best known algorithm for solving this problem is the "binary search" algorithm which takes $\log_2 N$ queries. In the paper [75] it is shown that a quantum computer solves the problem using $0.53 \log_2 N$ thus giving a constant speed-up from the classical algorithm [75]. [76] presents the best known deterministic quantum algorithm for this problem which uses $0.433 \log_2 N$ queries. A randomized quantum algorithm is given whose expected query complexity is less than $\frac{1}{3} \log_2 N$ [77]. Andrew Childs and Troy Lee showed that there is a lower bound of $\frac{\ln 2}{\pi} \log_2 N$ quantum queries for this problem [78].

Algorithm 2: Quantum Approximate Optimization Algorithm

Problem: We want to find the exact optimal solution to a combinatorial optimization problem as well as approximating it with a sufficiently small error.

Finding the exact optimal solution to a a combinatorial optimization problem is NP-complete, approximation with sufficiently small error bound is also NP-complete so this is clearly a challenging problem to solve classically. An algorithm known as Quantum Approximate Optimization Algorithm (QAOA) was proposed for finding approximate solutions to combinatorial optimization problems and offers a super-polynomial speedup to polynomial-time classical approximation algorithms [79]. The power of QAOA relative to classical computing is still a field of active research[80, 81, 82].

Algorithm 3: Gradient finding

Problem: Given a smooth function $f : R^d \to R$ we want to estimate ∇f at some specified point $f(x_0) \in R^d$.

A classical computer requires at least $d+1$ queries to solve the above mentioned problem whereas Stephen Jordan provides in [95], a quantum algorithm which can solve the problem in 1 step. In appendix D of his thesis [85] Stephen Jordan also shows that a quantum computer can use the gradient algorithm to find the minimum of a quadratic form in d dimensions using $O(d)$ queries, a problem for which a classical algorithm requires at least $O(d^2)$ queries [86]. These algorithms in general provide a polynomial speed-up over classical algorithms.

Algorithm 4: Semi-definite Programming

Problem: Given a list of $m + 1$ Hermitian $(n \times n)$ matrices $(C, A_1, A_2, \ldots, A_m)$ and m numbers $b_1, \ldots, b_m$, we may want to find the positive semi-definite $(n \times n)$ matrix X that maximizes $\mathrm{tr}(CX)$ subject to the constraints $(\mathrm{tr}(A_j X) \leq b_j)$ for $(j = 1, 2, \ldots, m)$ [92].

Quantum algorithms have been proposed which give polynomial and in special cases exponential speed-up. The paper [93] gives a quantum algorithm which approximately solves semi-definite problems to within $(\pm\delta)$ in time $(\widetilde{O}(n^{1/2}m^{1/2}R^{10}/\delta^{10}))$, where R is an upper bound on the trace of the solution which is a quadratic speed-up over the fastest classical algorithms. In the case where the input matrices have a sufficiently low rank a variant of the algorithm in [93] based on amplitude amplification and quantum Gibbs sampling can give an exponential speed-up.

Algorithm 5: Graph Properties in the Adjacency Matrix Model

Problem: Given a graph G with n vertices we wish to find out whether corresponding vertices are connected by an edge. These are problems in combinatorial optimization.

Quantum algorithms have been proposed which give a polynomial speed up in solving this problem. Given access to an oracle, which given a pair of integers in $1, 2, ..., n$ tells us whether the corresponding vertices are connected by an edge, the quantum query complexity of finding a minimum spanning tree of weighted graphs and deciding connectivity for directed and undirected graphs have $\Theta(n^{\frac{3}{2}})$ quantum query complexity and finding lowest weight paths has $O(n^{\frac{3}{2}}\log^2 n)$ quantum query complexity [19]. The paper [102] shows that deciding whether a graph is bipartite, detecting cycles and deciding whether a given vertex can be reached from another (st-connectivity) can all be achieved using a number of queries and quantum gates that both scale as $O(n^{\frac{3}{2}})$ and only logarithmically many qubits. Other interesting computational problems include detecting trees of a given size [103] and finding a subgraph H in a given graph G [104].

Mathematical optimization is also considered an important subdivision of machine learning and some information has been included in the appendix on machine learning and quantum machine learning, see Appendix D.

Chapter 5

Proposed quantum gradient algorithm

The task of the algorithm is to find the global minimum of a discrete convex function representing the optimal solution of a problem. The basic idea of the optimization algorithm is to combine a quantum version of the gradient search method with the Grover search algorithm. This can be seen as generating a new quantum search algorithm of a structured database given by the graph of the function. The resulting algorithm is faster than a pure Grover search and faster than a classical brute force search.

5.1 Working principle

We are searching for the minimum of a differentiable scalar field $f(x)$ with discrete values of x. We assume $f(x)$ to be a convex function. We begin by assuming that we can encode the $f(x)$ and x in quantum states. The algorithm then takes a position x encoded in a state vector as $|x\rangle$ and evolves the position state $|x_k\rangle$ into a different position state $|x_{k+1}\rangle$ defined on the search space, where k is an iteration number and $0 \leq k \leq 2^n$. This is accomplished by comparing the functional values in the neighbourhood of argument x and shifting/evolving the position state in the opposite direction of the gradient of the function f. If the iterative process is carried out using unitary operations, this will require additional registers. After a finite number of steps those states $|x\rangle$, where x being in the neighbourhood of local extrema are populated, such that the probability to detect local extrema upon measurement of the input register is increased. A further amplification of the amplitudes is achieved by carrying out a Grover search for states with gradient equal to the zero vector.

While we want to find local the minima of continuous functions $f : \mathbb{R}^d \to \mathbb{R}$, we assume here a discretized function $f : X^d \to X$ on a discrete d-dimensional grid X^d, where $X = \{1, 2, \dots N\}$. We define the components $g_i(\mathbf{x})$ of a discrete gradient $\mathbf{g}(\mathbf{x})$ by comparing the functional values of the arguments $\mathbf{x}$ and $\mathbf{x} + \mathbf{e_i}$ where $\mathbf{e_i}$ is the ith normalized basis vector:

$$g_i(\mathbf{x}) = \begin{cases} 1, & \text{for } f(\mathbf{x}) < f(\mathbf{x} + \mathbf{e}_i) \text{ and } f(\mathbf{x}) > f(\mathbf{x} - \mathbf{e}_i), \\ 0, & \text{for } f(\mathbf{x}) < f(\mathbf{x} + \mathbf{e}_i) \text{ and } f(\mathbf{x}) < f(\mathbf{x} - \mathbf{e}_i), \\ -1, & \text{for } f(\mathbf{x}) > f(\mathbf{x} + \mathbf{e}_i) \text{ and } f(\mathbf{x}) < f(\mathbf{x} - \mathbf{e}_i)\,, \end{cases} \tag{5.1}$$

and translate the state $|\mathbf{x}\rangle$ in direction of the negative discrete gradient:

$$|\mathbf{x}\rangle \to |\mathbf{x} - \mathbf{g}(\mathbf{x})\rangle\ . \tag{5.2}$$

In order to represent the purpose of the discrete gradient $\mathbf{g}(\mathbf{x})$ we denote it by $\nabla f(\mathbf{x})$. For simplicity one can interpret this problem as a graph search problem. A discrete problem is defined on a discrete state space $\mathbf{O}$. We can define a set of edges $x_i \in \mathbf{O}$ on the state space thus inducing a graph where i=(0,1,2...). A step by step search can then be implemented using functional values and the defined discrete gradient to find the state that minimizes a function taken over the graph.

5.2 Steps of the quantum gradient algorithm

The algorithm requires d input registers of length n and k output registers. Here d is the dimension of the convex function to be optimized and k the number of iterations needed for this purpose. An extra register is included in between the input registers and the auxiliary qubit, this is for the implementation of the Grover which will require the extra qubit. The first step in the quantum gradient descent estimation is to perform a Hadamard transform on the qubits in the input registers which creates a uniform superposition of states representing all points in the domain of the function.

$$|\mathbf{0}\rangle_I |\mathbf{0}\rangle |0\rangle_{0_1} \dots |0\rangle_{0_k} \xrightarrow{H^{\otimes n}\otimes\mathbb{1}\otimes\mathbb{1}_1\otimes\ldots\otimes\mathbb{1}_k} \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \left|\mathbf{x_0}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle |0\rangle_{0_1} \dots |0\rangle_{0_k} \ , \tag{5.3}$$

where the summation (superposition) is taken over all components of the vector $\mathbf{x}_0$, *i.e*, over the i^{th} positions (particles or nodes) $1 \leq i \leq 2^n$. The subscript k on $\mathbf{x_k}$ is an iteration number. Next, a gradient is taken over the superposition state $|\mathbf{x}\rangle$ by applying a unitary gate $U_{\nabla f}$ which takes the discrete gradient of the function and puts it in the auxiliary qubits. We also have an extra $|\mathbf{0}\rangle$ for the later implementation of the Grover algorithm.

$$\begin{aligned} &\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \left|\mathbf{x_0}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle |0\rangle_{0_1} \dots |0\rangle_{0_k} \\ &\xrightarrow{\mathbb{1}_I\otimes\mathbb{1}\otimes U_{\nabla f}\otimes\mathbb{1}_2\otimes\ldots\otimes\mathbb{1}_k} \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \left|\mathbf{x_0}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle |0\rangle_{0_2} \dots |0\rangle_{0_k} . \end{aligned} \tag{5.4}$$

The gradient is used to translate the state in the direction of the negative gradient using a unitary gate U_{step}

$$\begin{aligned} &\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \left|\mathbf{x_0}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle |0\rangle_{0_2} \dots |0\rangle_{0_k} \xrightarrow{(U_{\text{step}})_I\otimes\mathbb{1}\otimes\mathbb{1}_1\otimes\ldots\otimes\mathbb{1}_k} \\ &\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \left|\mathbf{x_0}^{(\mathbf{i})} - \nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle_I |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle |0\rangle_{0_2} \dots |0\rangle_{0_k} . \end{aligned} \tag{5.5}$$

The notation is simplified by setting $\mathbf{x_1}^{(\mathbf{i})} = \mathbf{x_0}^{(\mathbf{i})} - \nabla f(\mathbf{x_0}^{(\mathbf{i})})$. We proceed as in the gradient descent search algorithm to determine the gradient at the new position $x_1{}^{(i)}$ after the first iteration.

$$\vdots$$

after k iterations we get

$$\begin{aligned} &c_1 \sum_{i\notin \text{Min}} \left|\mathbf{x_{k-1}}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle \dots \left|\nabla f(\mathbf{x_{k-1}}^{(\mathbf{i})})\right\rangle + c_2 \sum_{i\in \text{Min}} \left|\mathbf{x_{k-1}}^{(\mathbf{i})}\right\rangle |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle \dots \left|\nabla f(\mathbf{x_{k-1}}^{(\mathbf{i})})\right\rangle \\ &\xrightarrow{(U_{\text{step}})} \\ &c_1 \sum_{i\notin \text{Min}} \left|\mathbf{x_k}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle \dots \left|\nabla f(\mathbf{x_k}^{(\mathbf{i})})\right\rangle + c_2 \sum_{i\in \text{Min}} \left|\mathbf{x_k}^{(\mathbf{i})}\right\rangle |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle \dots \left|\nabla f(\mathbf{x_{k-1}}^{(\mathbf{i})})\right\rangle \end{aligned} \tag{5.6}$$

where c_1 and c_2 are normalized coefficients where $c_1 = c_2 = \frac{1}{\sqrt{2^n}}$. Min is a set containing all points which are at most k steps away from the minimum value of the function. Here $i \notin$ Min refers to all points which are more than k steps away from the minimum, whereas $i \in$ Min refers to all points which are less than k steps away from the minimum. The unitary implementation which takes the gradient produces one of three outcomes which can be used to decide how to shift the vector towards a minimum.

The strength of the algorithm as is of many quantum algorithms is the use of the quantum superposition which allows these processes to be done in parallel. Taking the trace (calculate the reduced density matrix of the input) over the output registers we find that probability of measuring the input minimum state is $(\frac{2k+1}{2^n})^d$. In the above calculations, this amplitude amplification arises from normalization factors which arise in the converging towards the minimum. Intuitively one can think of all x positions which are within k steps of the minimum as converging at the minimum and hence boosting it's amplitude.

A subsequent application of the Grover search algorithm boosts the amplitude of the state corresponding to a vanishing gradient. In the below description operations on the auxiliary qubit during the Grover search are ignored without loss of generality for easier reading.

The first step in implementing the Grover algorithm is to implement a unitary operator called the oracle. The oracle marks the state corresponding to the minimum of the function with a "-1"

$$\begin{gathered} c_1 \sum_{i \notin \text{Min}} \left|\mathbf{x_k}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle ... \left|\nabla f(\mathbf{x_k}^{(\mathbf{i})})\right\rangle + c_2 \sum_{i \in \text{Min}} \left|\mathbf{x_k}^{(\mathrm{i})}\right\rangle |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle ... \left|\nabla f(\mathbf{x_{k-1}}^{(\mathbf{i})})\right\rangle \\ \xrightarrow{\text{Oracle}} c_1 \sum_{i \notin \text{Min}} \left|\mathbf{x_k}^{(\mathrm{i})}\right\rangle_I |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle ... \left|\nabla f(\mathbf{x_k}^{(\mathbf{i})})\right\rangle - c_2 \sum_{i \in \text{Min}} \left|\mathbf{x_k}^{(\mathrm{i})}\right\rangle |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle ... \left|\nabla f(\mathbf{x_{k-1}}^{(\mathbf{i})})\right\rangle \end{gathered} \tag{5.7}$$

The next step is do do an un-computing operation which reverses the $U_{\nabla f}$ and U_{step}, the reason for doing this is so that we can implement the diffusion operation in the next step which requires the original input states.

$$\begin{gathered} c_1 \sum_{i \notin \text{Min}} \left|\mathbf{x_k}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle ... \left|\nabla f(\mathbf{x_k}^{(\mathbf{i})})\right\rangle - c_2 \sum_{i \in \text{Min}} \left|\mathbf{x_k}^{(\mathrm{i})}\right\rangle |\mathbf{0}\rangle \left|\nabla f(\mathbf{x_0}^{(\mathbf{i})})\right\rangle ... \left|\nabla f(\mathbf{x_{k-1}}^{(\mathbf{i})})\right\rangle \\ \xrightarrow{\text{Uncompute}} c_1 \sum_{i \notin \text{Min}} \left|\mathbf{x_0}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle_{0_1} |\mathbf{0}\rangle_{0_2} ... |\mathbf{0}\rangle_{0_k} - c_2 \sum_{i \in \text{Min}} \left|\mathbf{x_0}^{(\mathbf{i})}\right\rangle |\mathbf{0}\rangle_{0_1} |\mathbf{0}\rangle_{0_2} \ldots |\mathbf{0}\rangle_{0_k} \end{gathered} \tag{5.8}$$

The next step in the Grover algorithm is to implement the diffusion operation which negates everything but the target states in the set, Min. This operation has been referred to in literature as an inversion about the mean, it has been detailed in chapter 2 of this thesis

$$\begin{gathered} c_1 \sum_{i \notin \text{Min}} \left|\mathbf{x_0}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle_{0_1} |\mathbf{0}\rangle_{0_2} ... |\mathbf{0}\rangle_{0_k} - c_2 \sum_{i \in \text{Min}} \left|\mathbf{x_0}^{(\mathbf{i})}\right\rangle |\mathbf{0}\rangle_{0_1} |\mathbf{0}\rangle_{0_2} \ldots |\mathbf{0}\rangle_{0_k} \\ \xrightarrow{\text{Diffusion}} c_1 \sum_{i \notin \text{min}} \left|\mathbf{x_0}^{(\mathbf{i})}\right\rangle_I |\mathbf{0}\rangle_{0_1} |\mathbf{0}\rangle_{0_2} ... |\mathbf{0}\rangle_{0_k} - c_2 \sum_{i \in \text{Min}} \left|\mathbf{x_0}^{(\mathbf{i})}\right\rangle |\mathbf{0}\rangle_{0_1} |\mathbf{0}\rangle_{0_2} \ldots |\mathbf{0}\rangle_{0_k}, \end{gathered} \tag{5.9}$$

We iteratively repeat the application of the Grover algorithm till we maximize the amplitudes in the range of Min, *i.e*, till we obtain the state which is approximately (See figure 5.1)

$$\sum_{i \in \text{Min}} \left|\mathbf{x_k}^{(\mathbf{i})}\right\rangle |\mathbf{0}\rangle_{0_1} |\mathbf{0}\rangle_{0_2} \ldots |\mathbf{0}\rangle_{0_k}. \tag{5.10}$$

However, in order to detect the position of the minimum, $\mathbf{x}^{(\mathbf{min})}$, one more run of the gradient search is needed. Lastly the minimum input state, $\left|\mathbf{x_k}^{(\mathbf{min})}\right\rangle$, is measured in the computational

Table 5.1: Table showing order of average number of repetitions for classical, Grover and quantum gradient search algorithms to find the minimum of a function.

Dimensions (d)	Grover plus Opt	Steps (Classical)	Grover
1	$O(N * k)^{1/2})$	$O(N)$	$O(N^{1/2})$
2	$O(N)$	$O(N^2/k)$	$O(N)$
3	$O(N^{3/2}/k^{1/2})$	$O(N^3/k^2)$	$O(N^{3/2})$
4	$O(N^2/k)$	$O(N^4/k^3)$	$O(N^2)$
5	$O(N^{5/2}/k^{3/2})$	$O(N^5/k^4)$	$O(N^{5/2})$

basis, *i.e*, on the input registers only. Taking the trace over the output gives us the probability of measuring the outcome of the input register. We calculate the number iterations to be of order, $O(\sqrt{N/(2k+1)})$, see number of iterations required for the Grover algorithm in chapter 2.

The analysis goes as follows, consider an ensemble of N points which are all set as initial points, where N is the size of the search space. If we allow all the points to move iteratively towards the minimum point, we find that after k iterations only the points which were within k steps of the minimum point actually converge to it. Hence the probability of finding a point at the maximum or the zero gradient after k steps through the set is of of order, $O(k/N)$. We can extend this to the d dimensional case where the probability of finding an optimal value in d dimensional space is given by order $O((k/N)^d)$ and hence the number of steps required has an upper bound of $O(N/k)^d$ steps for each iteration. For k iterations we have N^d/k^{d-1}. The Grover search algorithm has an upper bound of order, $O(N^{1/2})$ steps in searching an unstructured database. The upper bound for finding an optimal value in d dimensional space using only the Grover is $O(N^{d/2})$ steps. If we take the average number of repetitions to detect the minimum in the classical case to be of the order, $O(N/k)$ in the one dimensional case then combining this with the Grover would give an algorithm with an upper bound of order $O(N^{1/2}k^{-1/2})$ repetitions. We can generalize this to d dimensions to get $O(N^{d/2}/k^{d/2-1})$ average number of repetitions. Table 5.1 shows some analysis the upper bounds of three algorithms in increasing dimensions of the search function.

Table 5.1 shows that the quantum gradient search algorithm is always faster than its classical counterpart however in lower dimensions it is slower than a Grover algorithm. In higher dimensions the quantum gradient algorithm becomes faster than the Grover, from the table above we see that the quantum gradient search algorithm becomes faster when $d \geq 3$. A simple example of an implementation of the quantum gradient search algorithm is given below.

5.3 Example of an implementation of the the quantum gradient search algorithm on a basic one dimensional function

Consider a one dimensional function with 4 points such that $f(2) < f(1)$ and $f(2) < f(3)$ and $f(3) < f(4)$ making $f(2)$ the minimum. First we take the discrete gradient

$$\frac{1}{\sqrt{4}}(|\mathbf{1}\rangle + |\mathbf{2}\rangle + |\mathbf{3}\rangle + |\mathbf{4}\rangle)\,|0\rangle \xrightarrow{\text{grad}} \frac{1}{\sqrt{4}}(|\mathbf{1}\rangle\,|{-\mathbf{1}}\rangle_1 + |\mathbf{2}\rangle\,|\mathbf{0}\rangle_1 + |\mathbf{3}\rangle\,|\mathbf{1}\rangle_1 + |\mathbf{4}\rangle\,|\mathbf{1}\rangle_1). \tag{5.11}$$

We then take a step in search space

$$\frac{1}{\sqrt{4}}(|\mathbf{1}\rangle\,|{-\mathbf{1}}\rangle_1 + |\mathbf{2}\rangle\,|\mathbf{0}\rangle_1 + |\mathbf{3}\rangle\,|\mathbf{1}\rangle_1 + |\mathbf{4}\rangle\,|\mathbf{1}\rangle_1 \xrightarrow{\text{step}} \frac{1}{\sqrt{4}}(|\mathbf{2}\rangle\,|{-\mathbf{1}}\rangle_1 + |\mathbf{2}\rangle\,|\mathbf{0}\rangle_1 + |\mathbf{2}\rangle\,|\mathbf{1}\rangle_1 + |\mathbf{3}\rangle\,|\mathbf{1}\rangle_1). \tag{5.12}$$

We can simplify this to

$$\frac{1}{\sqrt{4}}(|\mathbf{2}\rangle\,|{-\mathbf{1}}\rangle_1 + |\mathbf{2}\rangle\,|\mathbf{0}\rangle_1 + |\mathbf{2}\rangle\,|\mathbf{1}\rangle_1 + |\mathbf{3}\rangle\,|\mathbf{1}\rangle_1) = \frac{1}{\sqrt{4}}\,|\mathbf{2}\rangle\,(|{-\mathbf{1}}\rangle_1 + |\mathbf{0}\rangle_1 + |\mathbf{1}\rangle_1) + \frac{1}{\sqrt{4}}\,|\mathbf{3}\rangle\,|\mathbf{1}\rangle_1\,. \tag{5.13}$$

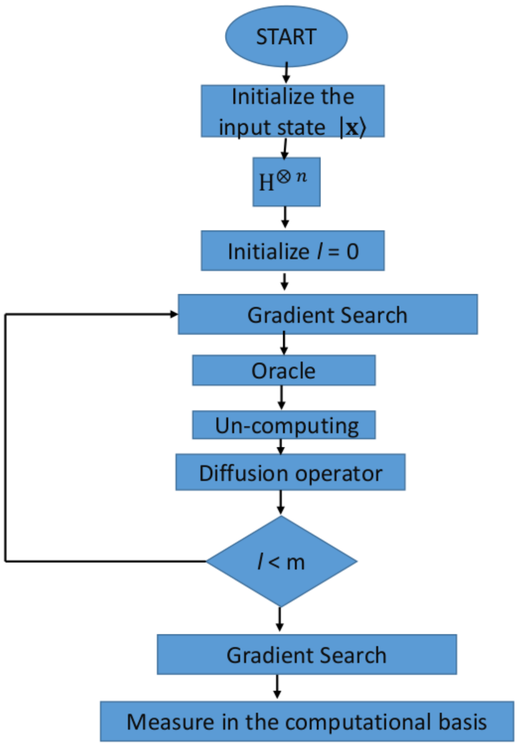


Figure 5.1: Flow diagram showing the working principle of the quantum gradient search algorithm. Here the number of Grover iterations, m, is of the order $O(\sqrt{N/(2k+1)})$.

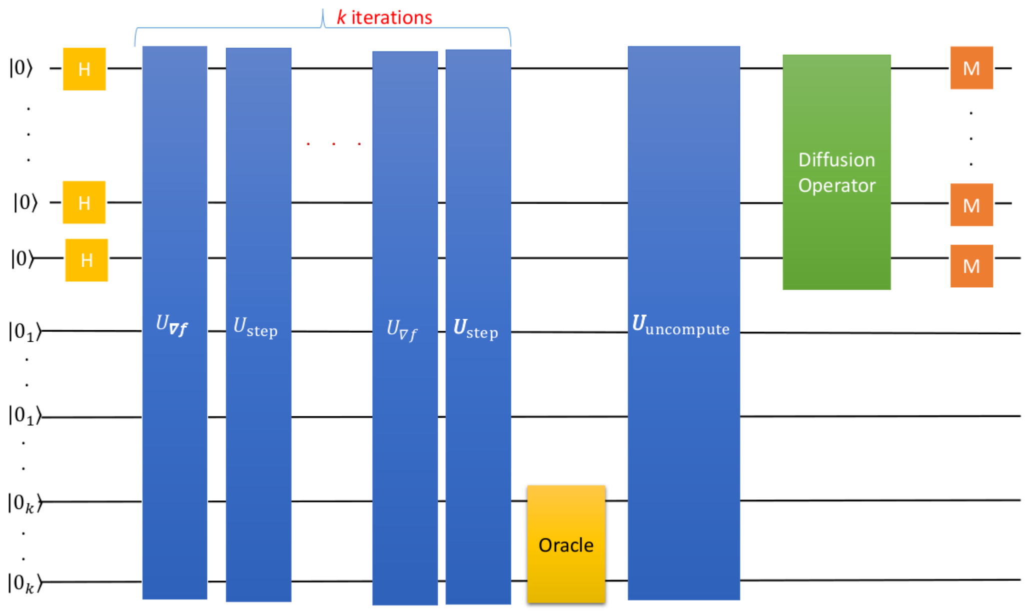


Figure 5.2: Quantum circuit showing the working principle of the quantum gradient search algorithm.

$$\frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left|-\mathbf{1}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{1}\right\rangle_1\right)+\frac{1}{\sqrt{4}}\left|\mathbf{3}\right\rangle\left|\mathbf{1}\right\rangle_1 \xrightarrow{\text{grad}} \frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left|-\mathbf{1}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{1}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2)+\frac{1}{\sqrt{4}}\left|\mathbf{3}\right\rangle\left|\mathbf{1}\right\rangle_1\left|\mathbf{1}\right\rangle_2. \tag{5.14}$$

we take another step

$$\frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left|-\mathbf{1}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{1}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\frac{1}{\sqrt{4}}\left|\mathbf{3}\right\rangle\left|\mathbf{1}\right\rangle_1\left|\mathbf{1}\right\rangle_2 \xrightarrow{\text{step}} \frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left|-\mathbf{1}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{1}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left|\mathbf{1}\right\rangle_1\left|\mathbf{1}\right\rangle_2. \tag{5.15}$$

we can reduce this to

$$\frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left|-\mathbf{1}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{1}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left|\mathbf{1}\right\rangle_1\left|\mathbf{1}\right\rangle_2=\frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left(\left|-\mathbf{1}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{1}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\left|\mathbf{1}\right\rangle_1\left|\mathbf{1}\right\rangle_2\right). \tag{5.16}$$

Oracle action marks the right state with a “ - ” so we have.

$$\frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left(\left|-\mathbf{1}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{1}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\left|\mathbf{1}\right\rangle_1\left|\mathbf{1}\right\rangle_2\right) \xrightarrow{\text{oracle}} -\frac{1}{\sqrt{4}}(\left|\mathbf{2}\right\rangle\left(\left(\left|-\mathbf{1}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{1}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\left|\mathbf{1}\right\rangle_1\left|\mathbf{1}\right\rangle_2\right), \tag{5.17}$$

$$-\frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left(\left|-\mathbf{1}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{1}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\left|\mathbf{1}\right\rangle_1\left|\mathbf{1}\right\rangle_2\right) \xrightarrow{\text{uncomputing}} \frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left(\left|\mathbf{0}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{0}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\left|\mathbf{0}\right\rangle_1\left|\mathbf{0}\right\rangle_2\right), \tag{5.18}$$

$$-\frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left(\left|\mathbf{0}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{0}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\left|\mathbf{0}\right\rangle_1\left|\mathbf{0}\right\rangle_2\right) \xrightarrow{\text{diffusion}} \frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left(\left|\mathbf{0}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{0}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\left|\mathbf{0}\right\rangle_1\left|\mathbf{0}\right\rangle_2\right). \tag{5.19}$$

The action of the diffusion operation is to mark everything but the target state. Lastly the state $\frac{1}{\sqrt{4}}\left|\mathbf{2}\right\rangle\left(\left|\mathbf{0}\right\rangle_1+\left|\mathbf{0}\right\rangle_1+\left|\mathbf{0}\right\rangle_1\right)\left|\mathbf{0}\right\rangle_2+\left|\mathbf{0}\right\rangle_1\left|\mathbf{0}\right\rangle_2)$ is measured in the computational basis, *i.e*, on the input registers only. Taking the trace over the output gives us the probability of measuring the outcome of the input register. It is clear that tracing over the output in the state in eqaution 5.14 we measure the outcome of the input register with probability $\frac{3}{4}$.

The above example is a very simplistic example to give proof of concept. In the above description operations on the auxiliary qubit during the grover search are ignored for the sake of clarity. In this exaggerated example we only do one grover iteration which wasn't even necessary here however this is not in general true.

Chapter 6

Proposed quantum Newton-Raphson algorithm to find the roots of a function

The task here is to find the roots of a scalar function $h : \mathbb{R} \to \mathbb{R}$. The basic idea of the root finding algorithm is to combine a quantum version of the Newton-Raphson method with the Grover search algorithm. This can be seen as generating a new quantum search algorithm of a structured database given by the graph of the function.

The resulting algorithm is faster than a pure Grover search and faster than the classical Newton-Raphson algorithm and its quantum version. The classical Newton-Raphson algorithm works by iterating in the direction of the ratio of the function with it's gradient that is $x_{i+1} = x_i - \frac{h(x)}{h'(x)}$ where i is the i^{th} iteration and $h(x)$ is the function whose roots we are searching for. Though the problem in this case is not necessarily an optimization problem the ideas used in this algorithm are quite similar to the ideas in the previous chapter.

6.1 Working principle

The proposed algorithm evolves a position state $|x\rangle$ to a different position state $|x'\rangle$ by comparing the functional values in the neighbourhood of argument x and shifting it into the direction of the ratio of the discrete function f with the discrete gradient of f given by $\frac{f(x)}{g(x)}$ where $f(x)$ and $g(x)$ are defined below. For this process to be carried out in a unitary manner it requires additional registers. The amplitude of the state $|x_{\text{root}}\rangle$ which correspond to the to the roots of the function increases with each step k.

After a finite number k of iterations we will have an increase in amplitude of the states associated with the roots of the function $|x_{\text{root}}\rangle$. As in the previous algorithm a further amplification of the amplitudes is achieved by carrying out a Grover search for states which correspond to the roots of the function. In this algorithm both the gradient and the function are discretized as shown below.

The gradient is defined as,

$$g(\mathbf{x}) = \begin{cases} 1, & \text{for}\, h(\mathbf{x}) < h(\mathbf{x}+\mathbf{e}) \text{ and } h(\mathbf{x}) > h(\mathbf{x}-\mathbf{e}), \\ 0, & \text{for}\;\, h(\mathbf{x}) \geq h(\mathbf{x}+\mathbf{e}) \text{ and } h(\mathbf{x}) \geq h(\mathbf{x}-\mathbf{e}), \\ -1, & \text{for}\, h(\mathbf{x}) > h(\mathbf{x}+\mathbf{e}) \text{ and } h(\mathbf{x}) < h(\mathbf{x}-\mathbf{e}_i)\,. \end{cases} \tag{6.1}$$

The function is defined as,

$$f(\mathbf{x}) = \begin{cases} 1, & \text{for}\, h(\mathbf{x}) > 0, \\ 0, & \text{for}\;\, h(\mathbf{x}) = 0, \\ -1, & \text{for}\, h(\mathbf{x}) < 0\ . \end{cases} \tag{6.2}$$

We condition the algorithm such that when $g(x) = 0$, then the point is not translated in either direction.

6.2 Steps of the root finding algorithm

As with the previous algorithm this algorithm requires d input registers of length n and k output registers. Here d is the number of the degree's of freedom of the function and k the number of iterations necessary to run the algorithm. The algorithm has 2 sets of k qubit auxiliary registers, one to store gradient values and another set for storing function values. An extra register is included in between the input registers and the auxiliary qubit registers, this is for the implementation of the Grover which will require the extra qubit.

The first step in this quantum version of the Newton-Raphson method is to perform a Hadamard transform over the input register creating a superposition over the x values of the function.

$$\begin{aligned}&|\mathbf{0}\rangle_I |0\rangle_{0_1} \dots |0\rangle_{0_k} |0\rangle_{0_1} \dots |0\rangle_{0_k} |0\rangle \xrightarrow{H^{\otimes n} \otimes \mathbb{1}_1 \otimes \dots \otimes \mathbb{1}_k \otimes \mathbb{1}_1 \otimes \dots \otimes \mathbb{1}_k \otimes \mathbb{1}} \\ &\sum_i a_i \left|{x_0}^i\right\rangle_I |0\rangle_{0_1} \dots |0\rangle_{0_k} |0\rangle_{0_1} \dots |0\rangle_{0_k} |0\rangle . \end{aligned} \tag{6.3}$$

where $a_i = \frac{1}{\sqrt{2^n}}$. The next step in the algorithm is to implement a unitary U_f which takes the function value at a point ${x_k}^i$.

$$\begin{aligned}&\sum_i a_i \left|{x_0}^i\right\rangle_I |0\rangle_{0_1} \dots |0\rangle_{0_k} |0\rangle_{0_1} \dots |0\rangle_{0_k} |0\rangle \xrightarrow{{\mathbb{1}_I}^{\otimes i} \otimes \mathbb{1}_1 \otimes \mathbb{1}_2 \otimes \dots \otimes \mathbb{1}_k \otimes U_f \otimes \mathbb{1}_2 \otimes \dots \otimes \mathbb{1}_k} \\ &\sum_i a_i \left|{x_0}^i\right\rangle_I |0\rangle_{0_1} \dots |0\rangle_{0_k} \left|f({x_0}^i)\right\rangle |0\rangle_{0_2} \dots |0\rangle_{0_k} |0\rangle . \end{aligned} \tag{6.4}$$

We then calculate the ratio of the function f with the gradient of f using a unitary gate U_g

$$\begin{aligned}&\sum_i a_i \left|{x_0}^i\right\rangle_I |0\rangle_{0_1} \dots |0\rangle_{0_k} |0\rangle_{0_1} \dots |0\rangle_{0_k} |0\rangle \xrightarrow{{\mathbb{1}_I}^{\otimes i} \otimes U_g \otimes \mathbb{1}_2 \otimes \dots \otimes \mathbb{1}_k \otimes \mathbb{1}_1 \otimes \mathbb{1}_2 \otimes \dots \otimes \mathbb{1}_k} \\ &\sum_i a_i \left|{x_0}^i\right\rangle_I \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \dots |0\rangle_{0_k} \left|f({x_0}^i)\right\rangle |0\rangle_{0_2} \dots |0\rangle_{0_k} |0\rangle . \end{aligned} \tag{6.5}$$

We have that root is a set containing all points which are k steps away from the root values of the function. Here $i \notin$ root refers to all points which are more than k steps away from the roots of the functions, whereas $i \in$ root refers to all points which are less than k steps within the range of the roots. For ease of notation we let $\mathbb{1}_{kk} = \mathbb{1}_1 \otimes \mathbb{1}_2 \otimes \dots \otimes \mathbb{1}_k$,

$$\begin{aligned}&\sum_i a_i \left|{x_0}^i\right\rangle_I \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \dots |0\rangle_{0_k} \left|f({x_0}^i)\right\rangle |0\rangle_{0_2} \dots |0\rangle_{0_k} |0\rangle \xrightarrow{(U_{\text{step}} \otimes \mathbb{1}_2 \otimes \dots \otimes \mathbb{1}_i)_I \otimes \mathbb{1}_{kk} \otimes \mathbb{1}_{kk}} \\ &\sum_i a_i \left|{x_0}^i - \frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_I \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \dots |0\rangle_{0_k} \left|f({x_0}^i)\right\rangle |0\rangle_{0_2} \dots |0\rangle_{0_k} |0\rangle . \end{aligned} \tag{6.6}$$

We simplify notation by setting ${x_1}^i = {x_0}^i - \frac{f({x_0}^i)}{g({x_0}^i)}$. After k iterations we have

$$\begin{aligned}&\sum_{i \notin \text{roots}} a_i \left|{x_k}^i\right\rangle \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \dots \left|\frac{f({x_{k-1}}^i)}{g({x_{k-1}}^i)}\right\rangle_{0_k} \left|f({x_0}^i)\right\rangle \dots \left|f({x_{k-1}}^i)\right\rangle_{0_k} |0\rangle \\ &+ \sum_{i \in \text{roots}} a_i \left|{x_k}^i\right\rangle \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \dots \left|\frac{f({x_{k-1}}^i)}{g({x_{k-1}}^i)}\right\rangle_{0_k} \left|f({x_0}^i)\right\rangle \dots \left|f({x_{k-1}}^i)\right\rangle_{0_k} |0\rangle . \end{aligned} \tag{6.7}$$

$$\xrightarrow{(U_{\text{step}}\otimes \mathbb{1}_2 \otimes \ldots \otimes \mathbb{1}_i)_I \otimes \mathbb{1}_{kk} \otimes \mathbb{1}_{kk}} \tag{6.8}$$

$$\begin{aligned}&\sum_{i\notin \text{roots}} a_i \left|{x_k}^i\right\rangle \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \cdots \left|\frac{f({x_{k-1}}^i)}{g({x_{k-1}}^i)}\right\rangle_{0_k} \left|f({x_0}^i)\right\rangle \ldots \left|f({x_{k-1}}^i)\right\rangle_{0_k} |0\rangle \\ &+ \sum_{i\in \text{roots}} a_i \left|{x_k}^i\right\rangle \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \cdots \left|\frac{f({x_{k-1}}^i)}{g({x_{k-1}}^i)}\right\rangle_{0_k} \left|f({x_0}^i)\right\rangle \ldots \left|f({x_{k-1}}^i)\right\rangle_{0_k} |0\rangle \,.\end{aligned} \tag{6.9}$$

As with the previous algorithm the next step is to implement the Grover search algorithm to maximize the amplitudes in the range of root. The next step after the iterative evolution of the position states inside the Grover algorithm is to mark the target root state. This is done using the unitary operator called the oracle

$$\begin{aligned}&\sum_{i\notin \text{roots}} a_i \left|{x_k}^i\right\rangle \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \cdots \left|\frac{f({x_{k-1}}^i)}{g({x_{k-1}}^i)}\right\rangle_{0_k} \left|f({x_0}^i)\right\rangle \ldots \left|f({x_{k-1}}^i)\right\rangle_{0_k} |0\rangle \\ &+ \sum_{i\in \text{roots}} a_i \left|{x_k}^i\right\rangle \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \cdots \left|\frac{f({x_{k-1}}^i)}{g({x_{k-1}}^i)}\right\rangle_{0_k} \left|f({x_0}^i)\right\rangle \ldots \left|f({x_{k-1}}^i)\right\rangle_{0_k} |0\rangle\end{aligned} \tag{6.10}$$

$$\xrightarrow{\text{Oracle}} \tag{6.11}$$

$$\begin{aligned}&\sum_{i\notin \text{roots}} a_i \left|{x_k}^i\right\rangle \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \cdots \left|\frac{f({x_{k-1}}^i)}{g({x_{k-1}}^i)}\right\rangle_{0_k} \left|f({x_0}^i)\right\rangle \ldots \left|f({x_{k-1}}^i)\right\rangle_{0_k} |0\rangle \\ &- \sum_{i\in \text{roots}} a_i \left|{x_k}^i\right\rangle \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \cdots \left|\frac{f({x_{k-1}}^i)}{g({x_{k-1}}^i)}\right\rangle_{0_k} \left|f({x_0}^i)\right\rangle \ldots \left|f({x_{k-1}}^i)\right\rangle_{0_k} |0\rangle \,.\end{aligned} \tag{6.12}$$

Then we perform the un-computing operation

$$\begin{aligned}&\sum_{i\notin \text{roots}} a_i \left|{x_k}^i\right\rangle \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \cdots \left|\frac{f({x_{k-1}}^i)}{g({x_{k-1}}^i)}\right\rangle_{0_k} \left|f({x_0}^i)\right\rangle \ldots \left|f({x_{k-1}}^i)\right\rangle_{0_k} |0\rangle \\ &- \sum_{i\in \text{roots}} a_i \left|{x_k}^i\right\rangle \left|\frac{f({x_0}^i)}{g({x_0}^i)}\right\rangle_{0_1} \cdots \left|\frac{f({x_{k-1}}^i)}{g({x_{k-1}}^i)}\right\rangle_{0_k} \left|f({x_0}^i)\right\rangle \ldots \left|f({x_{k-1}}^i)\right\rangle_{0_k} |0\rangle\end{aligned} \tag{6.13}$$

$$\xrightarrow{\text{un}-\text{computing}} \tag{6.14}$$

$$\begin{aligned}&\sum_{i\notin \text{roots}} a_i \left|{x_0}^i\right\rangle |\mathbf{0}\rangle_{0_1} \ldots |\mathbf{0}\rangle_{0_k} |\mathbf{0}\rangle \ldots |\mathbf{0}\rangle_{0_k} |0\rangle \\ &- \sum_{i\in \text{roots}} a_i \left|{x_0}^i\right\rangle |\mathbf{0}\rangle_{0_1} \ldots |\mathbf{0}\rangle_{0_k} |0\rangle \ldots |0\rangle_{0_k} |0\rangle \,.\end{aligned} \tag{6.15}$$

Table 6.1: Table showing order of number of steps for each algorithm

Dimensions (d)	Grover plus Opt	Steps (Classical)	Grover
1	$O(N * k)^{1/2}$	$O(N)$	$O(N^{1/2})$
2	$O(N)$	$O(N^2/k)$	$O(N)$
3	$O(N^{3/2}/k^{1/2})$	$O(N^3/k^2)$	$O(N^{3/2})$
4	$O(N^2/k)$	$O(N^4/k^3)$	$O(N^2)$
5	$O(N^{5/2}/k^{3/2})$	$O(N^5/k^4)$	$O(N^{5/2})$

Then the next step is to perform the diffusion operation

$$\begin{aligned}&\sum_{i\notin \text{roots}} a_i \left|{x_0}^i\right\rangle |\mathbf{0}\rangle_{0_1}\ldots|\mathbf{0}\rangle_{0_k}|\mathbf{0}\rangle\ldots|\mathbf{0}\rangle_{0_k}|0\rangle \\ &-\sum_{i\in \text{roots}} a_i \left|{x_0}^i\right\rangle |\mathbf{0}\rangle_{0_1}\ldots|\mathbf{0}\rangle_{0_k}|0\rangle\ldots|0\rangle_{0_k}|0\rangle \xrightarrow{\text{Diffusion}}\end{aligned} \tag{6.16}$$

$$\begin{aligned}&-\sum_{i\notin \text{roots}} a_i \left|{x_0}^i\right\rangle |\mathbf{0}\rangle_{0_1}\ldots|\mathbf{0}\rangle_{0_k}|\mathbf{0}\rangle\ldots|\mathbf{0}\rangle_{0_k}|0\rangle \\ &+\sum_{i\in \text{roots}} a_i \left|{x_0}^i\right\rangle |\mathbf{0}\rangle_{0_1}\ldots|\mathbf{0}\rangle_{0_k}|0\rangle\ldots|0\rangle_{0_k}|0\rangle .\end{aligned} \tag{6.17}$$

After some iterations we expect to measure any one of the root states with probability by tracing over the output,similar to what we do in the previous algorithm, *i.e*, the quantum gradient algorithm. It will be necessary to repeat measurement to get the other root states. Any one state associated with the roots can be written as being approximately equal to

$$\sum_{i\in \text{roots}} \left|{x_k}^i\right\rangle |\mathbf{0}\rangle_{0_1}\ldots|\mathbf{0}\rangle_{0_k}|0\rangle\ldots|0\rangle_{0_k}|0\rangle . \tag{6.18}$$

As with the previous gradient search algorithm it is necessary to repeat the number of Grover iterations as necessary to maximize the amplitude. The amplitude of the root states is bumped up by the grover iterations. Afterwards, one more iteration of the Newton-Raphson method is needed followed by a measurement in the computational basis of the first register. The complexity analysis is similar to that of the quantum gradient search algorithm. Consider the one dimensional case where we have N functional values and we allow k steps to be taken towards the root value by individual points representing all functional values then we will find that at most $O(k/N)$ of these points reach the roots hence the probability of finding a point at the roots or after k steps through the set is of order, $O(k/N)$. We can extend this to the d dimensional case such that the probability of finding a root value in d dimensional space is given by $O(k/N)^d$ and hence the number of steps required is has an upper bound of $O(N/k)^d$ steps for each iteration. For k iterations we need $O(N^d/k^{d-1})$ steps. The Grover search algorithm is said to have an upper bound of order $O(N^{1/2})$ steps in searching an unstructured database. The number of steps of finding root values in d dimensional space using only the Grover has an upper bound of $O(N^{d/2})$ steps. If we take the number of iterations in the classical case to be $O(N/k)$ in the one dimensional case then combining this with the Grover would give and algorithm with an upper bound given by $O(N^{1/2}/k^{1/2-1})$ steps. This follows from the fact that each iteration has an upper bound of $O(N^{1/2}/k^{1/2})$ and there are k steps hence we have an overall upper bound $O(N^{1/2}/k^{1/2-1})$ steps in one dimension. The upper bound for finding an optimal value in d dimensional space using the Grover and quantum optimisation algorithm is $O(N^{d/2}/k^{d/2-1})$

The analysis is the same as in the previous algorithm, we see that the quantum Newton-Raphson search algorithm is always faster than its classical counterpart however in lower dimensions it is slower than a Grover algorithm. In higher dimensions ($d \geq 3$) the quantum gradient algorithm becomes faster than the Grover however it should be noted that this comes at an expense in term of the amount of resources used. A simplified example is provided below to add clarity to how the algorithm actually works in practice.

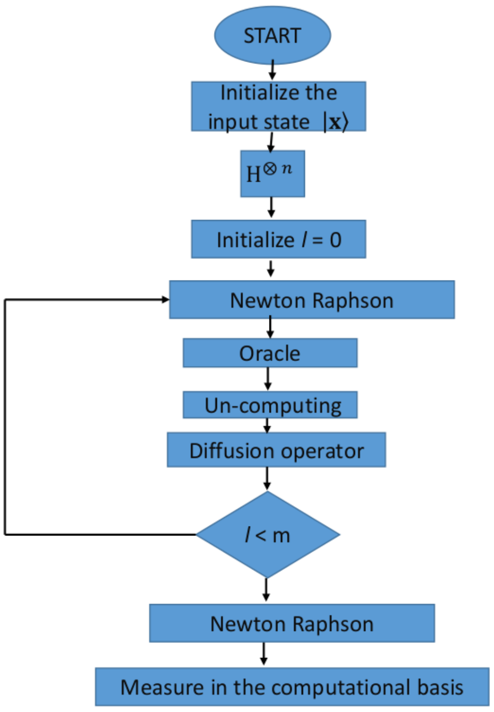


Figure 6.1: Flow diagram showing the working principle of the quantum Newton-Raphson algorithm. Here the number of Grover iterations, m, is of the order $O(\sqrt{N/(2k+1)})$.

6.3 Example of an implementation of the the quantum Newton-Raphson search algorithm on a basic one dimensional function

Consider a one dimensional function with 5 points such that $f(1) < 0$ and $f(2) = 0$ and $0 < f(3)$ and $f(4) = 0$ making $f(5) < 0$ where $f(2) = f(4) = 0$ are the roots of the function f. Implementing the quantum Newton-Raphson algorithm we find

$$\begin{aligned}&\frac{1}{\sqrt{5}}(|\mathbf{1}\rangle + |\mathbf{2}\rangle + |\mathbf{3}\rangle + |\mathbf{4}\rangle + |\mathbf{5}\rangle)\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle\\ &\xrightarrow{U_f} \frac{1}{\sqrt{5}}(|\mathbf{1}\rangle\,|\mathbf{0}\rangle\,|\mathbf{-1}\rangle + |\mathbf{2}\rangle\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle + |\mathbf{3}\rangle\,|\mathbf{0}\rangle\,|\mathbf{1}\rangle + |\mathbf{4}\rangle\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle + |\mathbf{5}\rangle\,|\mathbf{0}\rangle\,|\mathbf{1}\rangle).\end{aligned} \tag{6.19}$$

$$\begin{aligned}&\frac{1}{\sqrt{5}}(|\mathbf{1}\rangle\,|\mathbf{0}\rangle\,|\mathbf{-1}\rangle + |\mathbf{2}\rangle\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle + |\mathbf{3}\rangle\,|\mathbf{0}\rangle\,|\mathbf{1}\rangle + |\mathbf{4}\rangle\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle + |\mathbf{5}\rangle\,|\mathbf{0}\rangle\,|\mathbf{1}\rangle)\\ &\xrightarrow{U_{\nabla f}} \frac{1}{\sqrt{5}}(|\mathbf{1}\rangle\,|\mathbf{-1}\rangle\,|\mathbf{-1}\rangle + |\mathbf{2}\rangle\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle + |\mathbf{3}\rangle\,|\mathbf{-1}\rangle\,|\mathbf{1}\rangle + |\mathbf{4}\rangle\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle + |\mathbf{5}\rangle\,|\mathbf{1}\rangle\,|\mathbf{1}\rangle).\end{aligned} \tag{6.20}$$

We then take a step in search space

$$\begin{aligned}&\frac{1}{\sqrt{5}}(|\mathbf{1}\rangle\,|\mathbf{-1}\rangle\,|\mathbf{-1}\rangle + |\mathbf{2}\rangle\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle + |\mathbf{3}\rangle\,|\mathbf{-1}\rangle\,|\mathbf{1}\rangle + |\mathbf{4}\rangle\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle + |\mathbf{5}\rangle\,|\mathbf{-1}\rangle\,|\mathbf{1}\rangle)\\ &\xrightarrow{U_{\text{step}}} \frac{1}{\sqrt{5}}(|\mathbf{2}\rangle\,|\mathbf{-1}\rangle\,|\mathbf{-1}\rangle + |\mathbf{2}\rangle\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle + |\mathbf{4}\rangle\,|\mathbf{-1}\rangle\,|\mathbf{1}\rangle + |\mathbf{4}\rangle\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle + |\mathbf{4}\rangle\,|\mathbf{1}\rangle\,|\mathbf{1}\rangle).\end{aligned} \tag{6.21}$$

We can simplify this to

$$\frac{1}{\sqrt{5}}(|\mathbf{2}\rangle|\mathbf{-1}\rangle|\mathbf{-1}\rangle+|\mathbf{2}\rangle|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{4}\rangle|\mathbf{-1}\rangle|\mathbf{1}\rangle+|\mathbf{4}\rangle|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{4}\rangle|\mathbf{-1}\rangle|\mathbf{1}\rangle)=$$
$$\frac{1}{\sqrt{5}}|\mathbf{2}\rangle(|\mathbf{-1}\rangle|\mathbf{-1}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle)+\frac{1}{\sqrt{5}}|\mathbf{4}\rangle(|\mathbf{-1}\rangle|\mathbf{1}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{1}\rangle|\mathbf{1}\rangle). \tag{6.22}$$

In general we take k iterations but in this oversimplified example a single iteration was sufficient to get an observable increase in the the amplitudes of the states which correspond to roots of the function. In a more realistic data set which will have many more components we will remain with many points in the superposition which will have probabilities of being measured so we try to interfere then out so that we remain with high probability the correct states. In this example we leave out the extra grover register because it was not necessary to show the idea. In order to achieve this we implement the Grover search algorithm. The oracle of the Grover algorithm marks the right state with a "-" so we have.

$$\frac{1}{\sqrt{5}}|\mathbf{2}\rangle(|\mathbf{-1}\rangle|\mathbf{-1}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle)+\frac{1}{\sqrt{5}}|\mathbf{4}\rangle(|\mathbf{-1}\rangle|\mathbf{1}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{1}\rangle|\mathbf{1}\rangle)$$
$$\xrightarrow{\text{oracle}}-\frac{1}{\sqrt{5}}|\mathbf{2}\rangle(|\mathbf{-1}\rangle|\mathbf{-1}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle)-\frac{1}{\sqrt{5}}|\mathbf{4}\rangle(|\mathbf{-1}\rangle|\mathbf{1}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{1}\rangle|\mathbf{1}\rangle). \tag{6.23}$$

We then un-compute the action of the Newton-Raphson algorithm algorithm to get

$$-\frac{1}{\sqrt{5}}|\mathbf{2}\rangle(|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle)-\frac{1}{\sqrt{5}}|\mathbf{4}\rangle(|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle). \tag{6.24}$$

The next step is too perform the diffusion operation

$$-\frac{1}{\sqrt{5}}|\mathbf{2}\rangle(|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle)-\frac{1}{\sqrt{5}}|\mathbf{4}\rangle(|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle)\xrightarrow{\text{Diffusion}}$$
$$-\frac{1}{\sqrt{5}}|\mathbf{2}\rangle(|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle)-\frac{1}{\sqrt{5}}|\mathbf{4}\rangle(|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle). \tag{6.25}$$

The last step is to perform measurement in the measurement basis and get approximations to the states associated with the roots of the function. The state $-\frac{1}{\sqrt{5}}|\mathbf{2}\rangle(|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle)-\frac{1}{\sqrt{5}}|\mathbf{4}\rangle(|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle+|\mathbf{0}\rangle|\mathbf{0}\rangle)$ is measured in the computational basis, *i.e*, on the input registers only. Taking the trace over the output gives us the probability of measuring the outcome of the input register. It is clear that tracing over the output in the state in eqaution 6.22 we measure the outcome of the input register with probability $\frac{2}{5}$ for the root state $|2\rangle$ and with probability $\frac{3}{5}$ for the root state $|4\rangle$. In this example we only do one Grover iteration however this is not in general true.

Chapter 7

Conclusion and remarks

Quantum computation is very much still in it's early development phase despite the fact that people have been working on problems in the field for over 20 years. Quantum algorithms have been proposed which appear to be computationally faster than known classical algorithms. This has drawn much interest to the field. At present a major challenge comes in actually implementing some of these algorithms in the lab and scaling the implementations for high dimensionality calculations.

With the promise of quantum computers becoming a reality researchers have begun to design algorithms which solve a wide range of problems in numerical analysis and mathematical optimization. The problems solved using mathematical optimization are typically very important problems in everyday life.

Designing quantum algorithms is generally a difficult task. This is evident from the fact that there are so few viable quantum algorithms available despite the fact that the field of quantum computation has been growing for years. The goals of this thesis were to introduce previously proposed quantum optimization algorithms as well as new ones.

The second chapter of the thesis introduces some of the theory in quantum computation and goes in-depth about some of the most popular and more important results in the theory of quantum algorithms that is the Grover algorithm and the quantum factoring algorithm. These algorithms inspire the pursuit of quantum computation as a field of research with their promise of quadratic and exponential speed-up respectively over classical algorithms. In the third chapter an introduction of classical algorithms is made and the different techniques *i.e* iterative and heuristic methods used in solving mathematical optimization problems.

Examples of quantum algorithms used to solve mathematical optimization problems are presented in chapter 4. Chapter 4 summer details of the ordered search and quantum approximate algorithms amongst other optimization algorithms.

In the later chapters, *i.e*, chapters 5 and 6, I introduce the proposed algorithms, a quantum analogue of the gradient search algorithm, and a quantum analogue of the Newton-Raphson algorithm. The proposed quantum analogue of the gradient search algorithm and Newton-Raphson algorithm gives a quadratic speed-up over their classical counterparts.

Appendix A

Entanglement

Entangled states are combined states of qubits which exhibit correlations which are not observed in classical systems [127]. An example of these phenomenon is entanglement. The qubits are said to be entangled if their combined pure state is not separable. This means that their product cannot be written as a product of each of the individual qubits. Performing a measurement in one qubit of the entangled state instantaneously results in the state of the other qubit becoming defined (This happens even though each qubit does not have a definite state before measurement).

An example of an entangled system is the bipartite system given by

$$\left|\phi^{+}\right\rangle = \frac{1}{2}[|0\rangle_1 |0\rangle_2 + |1\rangle_1 |1\rangle_2] \tag{A.1}$$

where the subscripts refer to the individual qubits, the state is known as a Bell state. $|0\rangle$ and $|1\rangle$ are basis states of your Hilbert space. We can see that the state is entangled because it cannot be factorised or reduced into such a form as

$$\left|\phi^{+}\right\rangle = |0\rangle_1 \otimes |0\rangle_2 \tag{A.2}$$

This means we can no longer describe the system in terms of it's individual components but rather collective. Until a measurement is performed we cannot have information about each individual qubit. It is worth noting that by measuring the state of one qubit we can obtain the state of the other instantaneously. Another feature of entanglement is that it can occur between qubits/particles which are arbitrarily separated in space. In quantum optics a spontaneous parametric down conversion is used to create entangled photon pairs. Further reading on entanglement can be found in [115].

Appendix B

Adiabatic computing

There are many computationally interesting problems can be recast into an equivalent problem of finding a variable assignment that minimizes an “energy function[112]. Adiabatic quantum computation was first proposed by Farhi et al. as a method for solving NP-complete combinatorial optimization problems [112]. Adiabatic computing can be described as a form of quantum computing which relies on the adiabatic theorem to do calculations [112], it is a general approach to combinatorial search (study of search algorithms for solving instances of problems that are believed to be hard in general). It is a universal approach meaning that any computational problem can be expressed as an adiabatic algorithm. It is a contesting approach for quantum computer building and design. D-wave uses adiabatic algorithms for it's architecture. In adiabatic quantum computation one starts with an initial Hamiltonian, H_p(problem Hamiltonian), whose ground state describes the solution to the problem of interest, then a simple system, with Hamiltonian, $H(t)$, whose ground state is easy to prepare is prepared. The simple is (adiabatically) slowly varied to the Hamiltonian to one whose ground state encodes the solution to some computational problem. By the adiabatic theorem described here,

Assume you have a time dependant Hamiltonian $H(t)$ which has eigenvalues which vary in time as the Hamiltonian changes. One of them is the ground state energy $E_g(t)$. At any time there is a state known as the instantaneous ground state, *i.e*, the eigenstate of the Hamiltonian with eigenvalue $E_g(t)$.

If you imagine you have a a quantum mechanical system which you start at time t=0. The instantaneous ground state is given by

$$H(t)\left|E_g(t)\right\rangle = E_g(t)\left|E_g(t)\right\rangle . \tag{B.1}$$

Take the instantaneous ground state of the Hamiltonian and evolve with the Schrodinger equation

$$i\frac{d}{dt}\left|\psi(t)\right\rangle = H(t)\left|\psi(t)\right\rangle , \tag{B.2}$$

where

$$\left|\psi(0)\right\rangle = \left|E_g(t)\right\rangle . \tag{B.3}$$

The adiabatic theorem states that the state $|\psi(t)\rangle$ stays near $E_g(t)$ if H changes slowly enough. If I have a time independent Hamiltonian and I put the quantum system in the ground state then all that happens is that the phase changes and the system stays in the ground state. If the Hamiltonian changes slowly enough then you sweep the state along with the Hamiltonian. Slowness is determined by the spacing between the minimum excited state and the ground state.

Procedure for Implementing the Adiabatic algorithm :

Problem definition: Find the ground state of H_p which holds the solution to the problem

$$H_p\left|\psi\right\rangle = E(t)\left|\psi\right\rangle . \tag{B.4}$$

where $|\psi\rangle$ are eigenstates of the Hamiltonian. Inputs: Construct a Hamiltonian H(t) with an easy to construct ground state $|g\rangle$ and which in some basis which has the cost function in the diagonal. Steps: Initialize the Hamiltonian $H(t) = H_p$. Consider the $H(t) = (1 - t/T)H + (t/T)H_p$ where, T is the evolution time. We begin at t=0, so Hamiltonian evolves from $H(0) \rightarrow H_p$ by applying the Schrodinger equation

$$i\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle\,. \tag{B.5}$$

By the adiabatic theorem $|\psi(T)\rangle$ will be near the ground state of H_p if the evolution time, T, is big enough. Hence by the adiabatic theorem, the system will track the instantaneous ground state provided the variation of the Hamiltonian is sufficiently slow. It's runtime scales at worst as $\frac{1}{\gamma^3}$ where γ is defined as the minimum eigenvalue gap between the ground state and the first excited state [113]. A detailed analysis of the adiabatic computing can be found in [112].

Appendix C

Quantum annealing

Is defined as a meta-heuristic for finding the global minimum of an objective function given a set of candidate states (in which the solution exists)through using quantum fluctuations. Quantum fluctuations also known as quantum vacuum fluctuations or vacuum fluctuations are a temporary change in the energy content in a point space. This can be understood mathematically through the Werner Heisenberg principle given below

$$\Delta E \Delta t \geq \frac{\hbar}{4\pi} \tag{C.1}$$

where E is energy, t is time and $\hbar$ is the planck constant. Quantum annealing is an optimization techniques which employs quantum effects to escape the local minima of the objective function by tunneling through barriers separating local minima [69]. In the case where the cost function is really tall and has narrow barriers the quantum annealing algorithm can be more efficient than classical annealing which climbs over barriers.

Much of the popularity of the method stems from the fact that quantum annealing has been commercialized by D-wave (a Canadian company) in their devices. Their devices are analogue computation devices which are designed to use quantum annealing to solve an NP hard binary quadratic optimization problem with cost function $\sum_{i,j} a_{i,j} x_i x_j + \sum_i b_i x_i$, where the variables x_i can take the values 0 or 1.

While the quantum devices may not be able to solve NP-complete problems in polynomial time, they may have an advantage over classical algorithms which is yet to be explored. Hence the question as to whether quantum annealing is better than classical computing is still an open one.

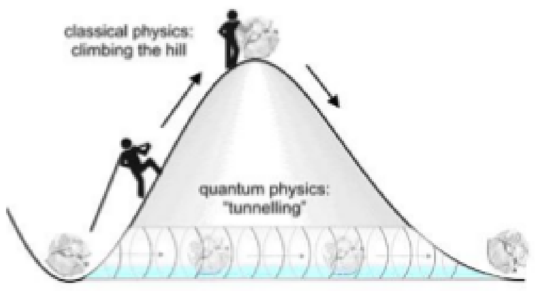


Figure C.1: Visual interpretation of quantum annealing [100]

Appendix D

Machine learning and quantum machine learning

The strong link between optimization problems and machine learning problems necessitate that a section on machine learning and quantum machine learning be included. Machine learning is a sub-field of machine learning which evolved from topic in Artificial Intelligence which include pattern recognition and computational learning theory. Peter Wittek defines machine learning more quantitatively as a field of Artificial intelligence that seeks patterns in empirical data without forcing models on the data [71]. A feature space which can be defined as a mathematical representation of the data instances under study, is at the heart of learning algorithms [71]. Learning models can be either statistical methods or they can be methods from algorithmic learning theory.

We employ machine learning in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult for example in email filtering, detection of network intruders or malicious insiders working towards a data breach [72], optical character recognition (OCR)[73], learning to rank and computer vision amongst many other applications.

Machine learning encompasses a wide variety of computational problems and can be attacked by a wide variety of algorithmic techniques. The problems in machine learning for which quantum algorithms have been devised include but are not limited to least squares fitting, binary classification, regression and cluster finding. Machine learning tasks are typically classified into three broad categories which are

Supervised Learning: These are learning tasks in which the algorithm is given example inputs and their desired outputs and the goal is for the machine learning algorithm to learn/devise a general rule which maps inputs to outputs.

Unsupervised Learning: These are learning tasks in which no labels are given to the learning algorithm.The machine learning algorithm is on its own in finding structure in its input. The goals of Unsupervised learning are discovering hidden patterns in data and feature learning.

Reinforcement Learning: These are learning tasks in which the machine learning algorithm interacts with a dynamic environment in which it must perform a certain goal. Examples of tasks performed include driving a vehicle or playing a game against an opponent. The learning algorithm is provided feedback in terms of rewards and punishments as it navigates its problem space.

These three categories typically cover all of the problems in machine learning. Machine learning tasks can further be classified depending on their desired output, *i.e*, as either classification tasks [114], regression tasks [116], clustering tasks [117], density estimation tasks or dimensionality reduction tasks [119].

There is such a great number of algorithmic techniques for solving a wide array of machine learning problems hence this section does not go into great detail but rather summarizes the techniques. A key algorithm in quantum machine learning is the HHL algorithm proposed by Seth Lloyd and colleagues [**?**]. The HHL algorithm has sparked much hope in quantum machine learning with it's promise for an exponential speed-up in certain cases. An example amongst many of an algorithm which uses a the HHL algorithm is quantum least squares fit algorithm which in certain conditions can give an exponential speed-up over the best classical algorithm was proposed [70].

Quantum machine learning (QML) is a new growing topic in machine learning. In some scholarly texts QML is defined as an intersection between quantum physics and machine learning. There are generally 4 ways to combine the two disciplines as shown in Figure D.1.

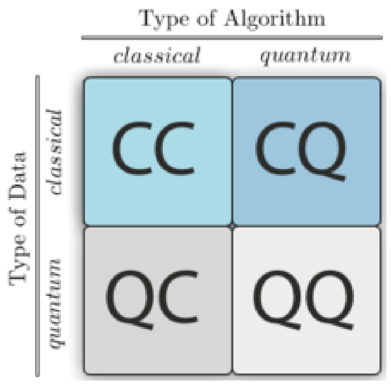


Figure D.1: Four different approaches to combine the disciplines of quantum computing and machine learning. Where the first letter refers to whether the system under study is classical or quantum, *i.e*, CC means the algorithm is classical and the data is also classical, while the second letter defines whether a classical or quantum information processing device is used [98].

Machine learning encompasses a wide variety of computational problems and quantum machine learning tries to extend the algorithms used here (classical algorithms) to quantum algorithms. The algorithms in quantum machine learning include quantum algorithms for solving linear systems [110] are applied to speed up cluster-finding, principal component analysis, binary classification and various forms of regression, provided the data satisfies certain conditions. In [111], these quantum algorithms for linear systems are applied to speed up the characterization of persistent homology from data sets.

Appendix E

Proposed quantum particle swarm optimization algorithm

In this chapter we present a preliminary result for an attempt at producing a quantum optimization algorithm which tries to find the minimum of a function $f(x)$ where $x \in R$. The algorithm takes from the particle swarm optimization algorithm and is a quantum analogue of it. We attempted to use multiple qubits in our search algorithm with the idea that having multiple qubits may enhance the searching capacity of the algorithm. In our pursuit toward a quantum particle swarm algorithm *i.e* an algorithm using quantum computation operations on a multiqubit system in order to resolve optimization problems, it became clear that a non-unitary operation was required to translate our system towards the state of the minimum of the function. We attempted to use a non-unitary operation known as the "Self Fulfilling Prophecy" [130].

E.1 Working principle

The proposed strategy performing operations required goes as follows. We begin with a multiqubit system of k qubits where $2 \leq k \leq N$ system which can be written as

$$|x_1\rangle \otimes |x_2\rangle \otimes |x_3\rangle \ldots |x_k\rangle . \tag{E.1}$$

The second step is to perform a Hadamard operation on each qubit and thus generate a a superposition of the position values x_i where $1 \leq i \leq 2^n$ associated with function i.e $\frac{1}{\sqrt{N}}\sum_{i=1,i\neq t}^{2^n-1} |x_i\rangle + \frac{1}{\sqrt{N}} |x_t\rangle$ where t is the target state. Here we explicitly write out the target state for emphasis. This can be generalized for each individual particle and we can write

$$\begin{aligned}&\Big(\frac{1}{\sqrt{N}}\sum_{i=1,i\neq t}^{2^n-1} |x_i\rangle + \frac{1}{\sqrt{N}} |x_t\rangle\Big)_1 \otimes \Big(\frac{1}{\sqrt{N}}\sum_{i=1,i\neq t}^{2^n-1} |x_i\rangle + \frac{1}{\sqrt{N}} |x_t\rangle\Big)_2 \ldots \\ &\otimes \Big(\frac{1}{\sqrt{N}}\sum_{i=1,i\neq t}^{2^n-1} |x_i\rangle + \frac{1}{\sqrt{N}} |x_t\rangle\Big)_k .\end{aligned} \tag{E.2}$$

In this case the target state is the state of the minimum of the function.

The third step in implementing the algorithm would be to introduce a unitary operation, $U_{\nabla f}$, which would take the gradient of each position. Thus we would generate another register to hold the gradient values $|\nabla f(x)\rangle$ which will contain a value $|1\rangle$ everywhere except at the state associated with the minimum of the function (the target state), where the gradient register will have be given by $|0\rangle$. The new state $|\Psi\rangle$ is written as

$$|\Psi\rangle = \Big(\frac{1}{\sqrt{N}}\sum_{i=1,i\neq t}^{2^n-1}|x_i\rangle|\nabla f(x_i)\rangle\rangle + \frac{1}{\sqrt{N}}|x_t\rangle|\nabla f(x_t)\rangle\rangle_1 \otimes \tag{E.3}$$
$$\Big(\frac{1}{\sqrt{N}}\sum_{i=1,i\neq t}^{2^n-1}|x_i\rangle|\nabla f(x_i)\rangle\rangle + \frac{1}{\sqrt{N}}|x_t\rangle|\nabla f(x_t)\rangle\rangle_2 \ldots$$

$$\otimes\Big(\frac{1}{\sqrt{N}}\sum_{i=1,i\neq t}^{2^n-1}|x_i\rangle|\nabla f(x_i)\rangle\rangle + \frac{1}{\sqrt{N}}|x_t\rangle|\nabla f(x_t)\rangle\rangle_k\ . \tag{E.4}$$

Since we can discretize our gradient such that $\nabla f(x_i) = 1$ for all states which are not the target state and $\nabla f(x_i) = 0$ for the target state we can rewrite $|\Psi\rangle$ as

$$|\Psi\rangle = \Big(\frac{1}{\sqrt{N}}\sum_{i=1\ and\ i\neq t}^{2^n-1}|x_i\rangle|1\rangle\rangle + \frac{1}{\sqrt{N}}|x_t\rangle|0\rangle\rangle_1 \otimes \tag{E.5}$$
$$\Big(\frac{1}{\sqrt{N}}\sum_{i=1\ and\ i\neq t}^{2^n-1}|x_i\rangle|1\rangle\rangle + \frac{1}{\sqrt{N}}|x_t\rangle|0\rangle\rangle_2 \ldots\otimes\Big(\frac{1}{\sqrt{N}}\sum_{i=1\ and\ i\neq t}^{2^n-1}|x_i\rangle|1\rangle\rangle + \frac{1}{\sqrt{N}}|x_t\rangle|0\rangle\rangle_k\ .$$

The method is quite similar to using an oracle. We can further generalize $|\Psi\rangle$ to $\beta|x_i\rangle|1\rangle + \alpha|x_t\rangle|0\rangle$ where β is the probability of measuring any other state but the target state and α is the probability of measuring the target state and thus the above state $|\Psi\rangle$ can again be written as

$$|\Psi\rangle = (\beta|x_i\rangle|1\rangle + \alpha|x_t\rangle|0\rangle)_1 \otimes (\beta|x_i\rangle|1\rangle + \alpha|x_t\rangle|0\rangle)_2 \ldots \otimes (\beta|x_i\rangle|1\rangle + \alpha|x_t\rangle|0\rangle)_k \tag{E.6}$$

In the above expression $\beta > \alpha$, this is because β is the probability of measuring any one of $N-1$ states out of N possible whereas α is only the probability of measuring one state of the N possible. Since the goal of the algorithm is to measure the target state with high probability, *i.e*, measure $|x_t\rangle$ with high probability, we introduce a non-unitary approach to translate the entire system to the target state.

The fourth step in the algorithm is to introduce the non-unitary and evolve the system towards the target state. The proposed non-unitary operation is known as the Self-fulfilling prophecy [130]. The aim in introducing this method can be shown as follows

$$|\Psi\rangle \xrightarrow{\text{SFP}} (\beta|x_t\rangle|1\rangle + \alpha|x_t\rangle|0\rangle)_1 \otimes (\beta|x_t\rangle|1\rangle + \alpha|x_t\rangle|0\rangle)_2 \ldots \otimes (\beta|x_t\rangle|1\rangle + \alpha|x_t\rangle|0\rangle)_k \tag{E.7}$$

Hence the final state is

$$((\beta+\alpha)|x_t\rangle(|1\rangle+|0\rangle))_1 \otimes ((\beta+\alpha)|x_t\rangle(|1\rangle+|0\rangle))_2 \ldots \otimes ((\beta+\alpha)|x_t\rangle(|1\rangle+|0\rangle))_n \tag{E.8}$$

We would thus be able to measure the target states $|x_t\rangle_1 \otimes |x_t\rangle_2 \ldots |x_t\rangle_n$ with a very high probability of $|(\beta+\alpha)|^2$. We ran some simulations to confirm the viability of the method claimed here and some preliminary results are presented below.

E.2 Simple case test for the proposed method

Since $\langle x_i|x_t\rangle = 0$, *i.e*, the states are orthogonal we can generalize the above expression as $|\Psi_G\rangle = \beta|1\rangle|1\rangle + \alpha|0\rangle|0\rangle$ where α and β are amplitudes of the target state and all other states respectively.

Now having the state in this form one can imagine that it is possible to evolve the above state $|\Psi_G\rangle$ in such a way that the amplitude associated with the target state can be amplified as in the Grover search. A unitary approach cannot achieve this as it would lead to a hard measurement of the states.

This is not ideal because the probability of measuring any other state is $\frac{N-1}{N}$ hence measuring the actual target state is very unlikely, rendering the algorithm inefficient. A non-unitary approach known as the Self-fulfilling prophecy (SFP) is introduced to overcome the problem. The SFP is conditioned not to change the state associated with the zero gradient.

The Self-fulfilling prophecy is a mechanism of preparing finite-dimensional quantum systems into target states. The mechanism works by means of unsharp measurements combined with feedback which aims to restore a presumed pre-measurement quantum state. This combination of measurements and feedback deterministically drives any state to a common state. Details of the SFP can be found in [130]. The SFP can only be applied to the first register. Beginning with the state Ψ_G , the protocol for SFP is as follows.

$$\mathrm{SFP}\,|\Psi_G\rangle = \alpha((\mathrm{SFP}\otimes \mathbb{1})\,|0\rangle \otimes |0\rangle) + \beta((\mathrm{SFP}\otimes \mathbb{1})\,|1\rangle \otimes |1\rangle) \tag{E.9}$$

The goal in doing this was that we could let the state $\beta((\mathrm{SFP}\otimes \mathbb{1})\,|1\rangle \otimes |1\rangle)$ evolve to $\beta(|0\rangle \otimes |1\rangle)$.

E.2.1 Implementing the working principle

The matlab code in the Appendix H was used to implement the proposed algorithm. It measures the fidelity of the outcome state after performing the SFP with the target state. The goal was to get a fidelity as close to 1 as possible. The outcome is shown in the Figure E.1 below

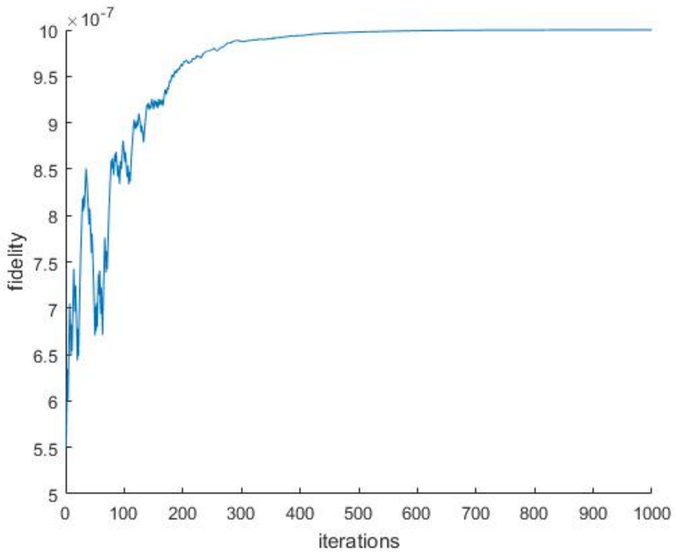


Figure E.1: Picture showing the fidelity of the state $\alpha\frac{1}{\sqrt{2}}\,|0\rangle\,|0\rangle + \beta\frac{1}{\sqrt{2}}\,|0_x\rangle\,|1\rangle$ against the $|0\rangle\,|0\rangle$ state under the SFP principle

Clearly the proposed method does not lead to the desired result as the fidelity of the outcome state with the target state is a very low value, much lower than 1.

E.3 Results and conclusion

Simulations were conducted in Matlab, the code which is given in Appendix F. We found that using SFP did not lead us to the desired state and inevitably we could get a gain in amplitude. The reason for this was could be because the SFP was not the most appropriate non-unitary technique to evolve our state towards the target state. The amplitudes may also play a pivotal role, for example the very small α value could easily be approximated as 0 in the matlab code thus destroying information about the vector associated with it. Though the method did not lead to the desired result more work is still being conducted in search of alternative methods.

Appendix F

Matlab code used in QPSO

F.0.1 Implementing SFP

In order to implement the SFP we had to generate the measurement matrices M_+ and M_- and the unitary matrices U_+ and U_-

$$M_+ = \begin{bmatrix} \sqrt{p_o} & 0 \\ 0 & \sqrt{1-p_o} \end{bmatrix} \tag{F.1}$$

$$M_- = \begin{bmatrix} \sqrt{1-p_o} & 0 \\ 0 & \sqrt{p_o} \end{bmatrix} \tag{F.2}$$

and

$$U_+ = \begin{bmatrix} \frac{\sqrt{p_o}+\sqrt{1-p_o}}{\sqrt{2}} & \frac{\sqrt{1-p_o}+\sqrt{p_o}}{\sqrt{2}} \\ -\frac{\sqrt{1-p_o}+\sqrt{p_o}}{\sqrt{2}} & \frac{\sqrt{p_o}+\sqrt{1-p_o}}{\sqrt{2}} \end{bmatrix} \tag{F.3}$$

$$U_- = \begin{bmatrix} \frac{\sqrt{p_o}+\sqrt{1-p_o}}{\sqrt{2}} & -\frac{\sqrt{1-p_o}+\sqrt{p_o}}{\sqrt{2}} \\ \frac{\sqrt{1-p_o}+\sqrt{p_o}}{\sqrt{2}} & \frac{\sqrt{p_o}+\sqrt{1-p_o}}{\sqrt{2}} \end{bmatrix} \tag{F.4}$$

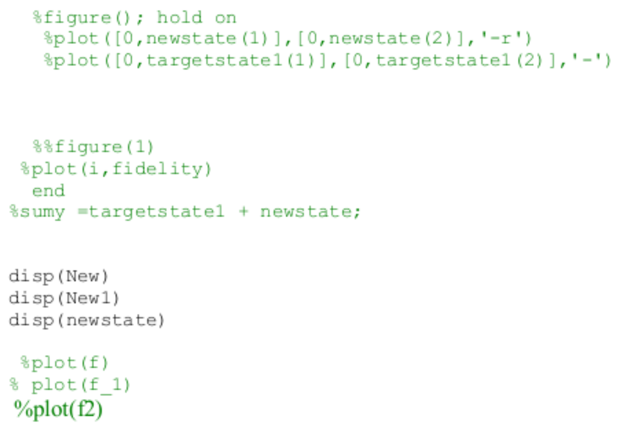
The code used to run the simulations presented in Appendix E has been added here

```
\begin{equation}
%for p_o = 0:0.1:1
%disp(p_o);
%close all;
%clc;
%clear all;
%counts=0;
p_o=0.45;
f=zeros(1,1000);
f_1=zeros(1,1000);
f2=zeros(1,1000);
N=512;
%INITIALIZING THE UNITARY AND MEASUREMENT MATRICES
M_P = [sqrt(p_o) 0; 0 sqrt(1-p_o)];
M_N = [sqrt(1-p_o) 0; 0 sqrt(p_o)];

amplitude=1/sqrt(2);
Comp1=amplitude*(sqrt(p_o) + sqrt(1-p_o));
Comp2 =amplitude*(sqrt(1-p_o) - sqrt(p_o));

U_P=[Comp1 Comp2;-Comp2 Comp1]
U_N=[Comp1 -Comp2;Comp2 Comp1]

initialstate = [0; 1];
initialstate1 = [1; 0];

alpha= 1/sqrt(N);
beta = sqrt(N-1)/sqrt(N);

%alpha= 1;
%beta =1/sqrt(10);

zeroxsate=[1/sqrt(2) 1/sqrt(2)];
optTARGETstate=[1 0];
superpositionstate=[0 1];



%TENSOR PRODUCT
%probability= transpose(initialstate)* conj(M_P)*M_P*initialstate;
%disp(probability)
%PREPARING THE INITIAL STATE

targetstate = [amplitude ; amplitude];

targetstate1= alpha*kron(zeroxsate,optTARGETstate);
targetstate2= alpha*kron(zeroxsate,optTARGETstate)+
beta*kron(zeroxsate,superpositionstate);
targetstate3=beta*kron(zeroxsate,superpositionstate);
targetstate4=beta*kron(superpositionstate,-zeroxsate);
targetstate5= alpha*kron(zeroxsate,optTARGETstate)-
beta*kron(zeroxsate,superpositionstate);
%disp(r);
New = initialstate
New1 = initialstate1
%if( Rand <0.5)


%GENERATING ITERATIONS OF THE MEASUREMENT AND UNITARY(WHEN p_o < 0.5)
%R = U_N*M_N*U_P*M_P;
```

Figure F.1: Matlab code for QPSO pg 1

```
%figure(1)
%figure(2)
% xlabel('iterations')
%ylabel('fidelity')

 hold on
  for i = 1: 1: 1000
 %disp(i)
   %figure(); hold on
   % plot([0,targetstate(1)],[0,targetstate(2)],'-')
   % plot([0,New(1)],[0,New(2)],'-r')
  r =rand()
   %New = New
   probability= transpose(New)* conj(M_P)*M_P*New;

  %if (mod(i,2) ~= 0)
      %disp(['prob']);
         disp([probability])
      if( r < probability)

      R = U_P*M_P*New
      R1 = U_P*M_P*New1
      New = R
      New1 = R1
      New =New./sqrt((New.'*New));
      New1 =New1./sqrt((New1.'*New1));
      disp(New/(New.'*New))
      disp(New1/(New1.'*New1))
   % disp('i ran case 1')
    %plot([0,New(1)],[0,New(2)],'-')
        else


      R = U_N*M_N*New
      R1 = U_N*M_N*New1
      New = R
      New1 = R1
      New =New./sqrt((New.'*New));
      New1 =New1./sqrt((New1.'*New1));
      disp(New/(New.'*New))
      disp(New1/(New1.'*New1))
    %disp('i ran case 1')
    end
    %ans=transpose(R)*initialstate

    newstate = alpha*kron(transpose(New),optTARGETstate) -
beta*kron(transpose(New1),superpositionstate);
    disp(newstate);
   % newstate=newstate/(newstate.'*newstate)
    fidelity1=(newstate*transpose(targetstate3))^2;
    f_1(i)=fidelity1;
    disp(f_1(1000))


    fidelity=(transpose(New)*targetstate)^2;
    disp(fidelity)
    f(i)=fidelity;

    fidelity2=(transpose(New1)*targetstate)^2;
    disp(fidelity2)
    f2(i)=fidelity;
```

Figure F.2: Matlab code for QPSO pg 2



```
  %figure(); hold on
   %plot([0,newstate(1)],[0,newstate(2)],'-r')
   %plot([0,targetstate1(1)],[0,targetstate1(2)],'-')


  %%figure(1)
 %plot(i,fidelity)
  end
%sumy =targetstate1 + newstate;


disp(New)
disp(New1)
disp(newstate)

 %plot(f)
% plot(f_1)
 %plot(f2)
```

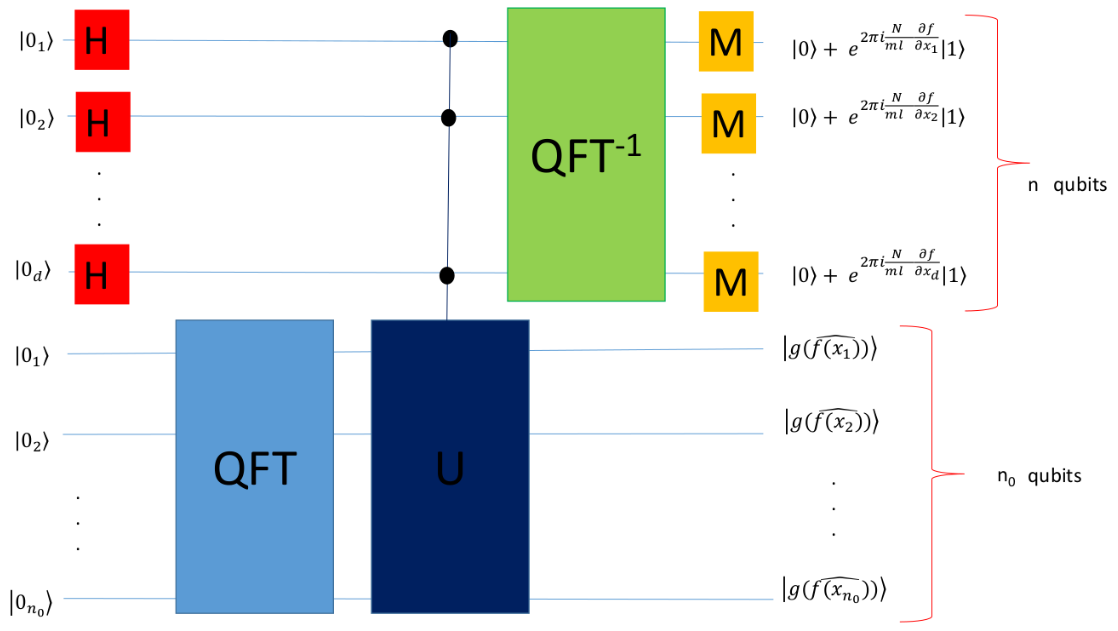
Figure F.3: Matlab code for QPSO pg 3

Appendix G

Quantum numerical gradient estimation algorithm

There is a grave number of applications for quantum algorithms [84]. A nice example of a quantum algorithm which can be used for optimization purposes is given in the paper by Stephen Jordan [95]. This algorithm is a generalization of the Bernstein-Vazirani algorithm [99].

In his paper Stephen Jordan tries to use the methods of the Bernstein-Vazirani algorithm to approximate the numerical gradient of a real valued function. In the quantum algorithm it suffices to show how to perform a quantum gradient estimation at $\mathbf{x} = \mathbf{0}$, since the gradient at other points can be obtained by trivially redefining f [95]. The algorithm uses a black box which performs a controlled-U operation given by $C - U : |x\rangle |a\rangle = |x\rangle |a \oplus g(x)\rangle$. The algorithm is designed to estimate the gradient $\nabla f(x)$ at a given point $(x = x_1, x_2, \ldots, x_d)$ with n bits of precision.



$|0_1\rangle$
$|0_2\rangle$
$|0_d\rangle$
H
H
H
QFT^{-1}
M
M
M
$|0\rangle + e^{2\pi i \frac{N}{ml}\frac{\partial f}{\partial x_1}}|1\rangle$
$|0\rangle + e^{2\pi i \frac{N}{ml}\frac{\partial f}{\partial x_2}}|1\rangle$
$|0\rangle + e^{2\pi i \frac{N}{ml}\frac{\partial f}{\partial x_d}}|1\rangle$
n qubits
$|0_1\rangle$
$|0_2\rangle$
$|0_{n_0}\rangle$
QFT
U
$|g(\widehat{f(x_1)})\rangle$
$|g(\widehat{f(x_2)})\rangle$
$|g(\widehat{f(x_{n_0})})\rangle$
n_0 qubits

Figure G.1: Quantum circuit for the implementation of a numerical gradient estimation algorithm.

Applying the unitary performs the operation $C - U : |x\rangle |a\rangle = |x\rangle \left|a \oplus (\frac{NN_0}{ml})g(x) \text{mod } N_0\right\rangle$. It can be shown that the eigenstate of the state $|g(x)\rangle$ is $\left|g(\hat{f(x)})\right\rangle$ under the transformation $|g(x)\rangle = \sum_{f(x)=0}^{2^n-1} e^{2\pi i(\frac{NN_0}{ml})xf(x)} \left|g(\hat{f(x)})\right\rangle$. This is similar to the period finding algorithm where we approximate $\frac{\hat{s}}{l}$ as some function $f(x)$. The function $f(x)$ can be approximated as $f(\frac{l}{N}(x - \frac{N}{2}))$, where N is the d-dimensional vector $(N_1, N_2, \ldots, N_d)$, and l is the size of the region over which the function f is approximately linear, m is the size of the interval which bounds the components of ∇f which ensure proper scaling of the final result into a fixed point representation (A fixed point number has a specific number of bits (or digits) reserved for the integer part (the part to the left of the decimal point) and a specific number of bits reserved for the fractional part (the part to the right of the decimal point). It is unlike a floating point point number which allows for a varying number of digits after the decimal point), that is, as an integer from 0 to 2^n [95] and $0 \leq N_0 \leq 2^{n_0 - 1}$. The algorithm provides a polynomial speed-up in gradient search over it's classical counterparts. The algorithm uses a black box which computes f and initializes d qubits of length n (where d is the dimension (degrees of freedom) of the function) and n_0 auxiliary qubits.

The initial states are $|0\rangle^{\otimes d}|0\rangle^{\otimes n_0}$. A Hadamard operation is then performed which produces a superposition of states represented as

$$\xrightarrow{\mathrm{H^{\otimes d}} \otimes \mathbb{1}^{\otimes n_0}} (\frac{1}{\sqrt{2}})^d (|0\rangle + |1\rangle)^{\otimes d}|0\rangle^{\otimes n_0} \tag{G.1}$$

$$\xrightarrow{\mathbb{1}^{\otimes \mathrm{d}} \otimes \mathrm{QFT}} (\frac{1}{\sqrt{2}})^d (|0\rangle + |1\rangle)^{\otimes d} \sum_{a=0}^{N_0 - 1} e^{2\pi i a/N_0} |a\rangle \tag{G.2}$$

We can simply $(\frac{1}{\sqrt{2}})^d (|0\rangle + |1\rangle)^{\otimes d} = \sum_{x=0}^{2^d - 1} |x\rangle$ where $x = (x_1, x_2, \ldots, x_d)$, $a = 0$, and it follows that after applying the quantum Fourier transform on the second register that we get

$$\frac{1}{\sqrt{2^d N_0}} \sum_{x=0}^{2^d - 1} |x\rangle \sum_{a=0}^{N_0 - 1} e^{2\pi i a/N_0} |a\rangle \tag{G.3}$$

The next step is to apply above mentioned unitary operator $\mathrm{C - U}$ on the second register which gives the new phase shown below

$$\xrightarrow{\mathbb{1}^{\otimes \mathrm{d}} \otimes \mathrm{U}^{\otimes \mathrm{n_0}}} \frac{1}{\sqrt{2^d N_0}} \sum_{x=0}^{2^d - 1} |x\rangle \sum_{a=0}^{N_0 - 1} e^{2\pi i a/N_0} |g(x)\rangle \tag{G.4}$$

substituting $|g(x)\rangle$ with it's eigenvalue approximation we get

$$\frac{1}{\sqrt{2^d N_0}} \sum_{x=0}^{2^d - 1} |x\rangle \sum_{f(x)=0}^{2^n - 1} e^{2\pi i (\frac{N N_0}{ml}) x f(x)} \left| g(\hat{f(x)}) \right\rangle \tag{G.5}$$

We then substitute in the above mentioned approximation to f, $f(x) \approx f(\frac{l}{N}(x - \frac{N}{2}))$, we get

$$\frac{1}{\sqrt{2^d N_0}} \sum_{x=0}^{2^d - 1} \sum_{f(x)=0}^{2^n - 1} e^{2\pi i \frac{N}{ml} x f(\frac{l}{N}(\delta - \frac{N}{2}))/N_0} |x\rangle \left| g(\hat{f(x)}) \right\rangle \tag{G.6}$$

If l is the diameter of the neighbourhood around which the function f is approximately linear then f can be we can approximated by means of a first derivative as shown below

$$\approx \frac{1}{\sqrt{2^d N_0}} \sum_{x=0}^{2^d - 1} \sum_{f(x)=0}^{2^n - 1} e^{i 2\pi \frac{N}{ml}(f(0) + (x - \frac{N}{2}).\nabla f)} |x\rangle \left| g(\hat{f(x)}) \right\rangle \tag{G.7}$$

Ignoring the global phase we can approximate the states to

$$\approx \frac{1}{\sqrt{2^d}} \sum_{x=0}^{2^d - 1} \sum_{f(x)=0}^{2^n - 1} e^{i 2\pi \frac{N}{ml}(\delta_1 \frac{\partial f}{\partial x_1} + x_2 \frac{\partial f}{\partial x_2} \ldots + x_d \frac{\partial f}{\partial x_d})} |x_1\rangle |x_2\rangle \ldots |x_d\rangle \left| g(\hat{f(x)}) \right\rangle \tag{G.8}$$

This can be expressed in the product state as

$$= \frac{1}{\sqrt{2^d}} \sum_{f(x)=0}^{2^n-1} (\sum_{\delta=0}^{2^d-1} e^{i2\pi \frac{N}{ml}(x_1 \frac{\partial f}{\partial x_1})} |x\rangle) \dots (\sum_{x} e^{i2\pi \frac{N}{ml}(x_d \frac{\partial f}{\partial x_d})} |x_d\rangle) \left| g(\hat{f(x)}) \right\rangle \tag{G.9}$$

Taking a single component of the above equation,

$$\frac{1}{\sqrt{2^d}} \sum_{f(x)=0}^{2^n-1} \sum_{\delta=0}^{2^d-1} e^{2\pi i \frac{N}{ml}(x_1 \frac{\partial f}{\partial x_1})} |x_1\rangle \left| g(\hat{f(x_1)}) \right\rangle \tag{G.10}$$

Thinking back to the order-finding algorithm we can see that one can look at $\frac{N}{ml}(\frac{\partial f}{\partial x_1})$ as equivalent to $\frac{\hat{s}}{r}$. The next step in the algorithm is to implement the inverse QFT, and it becomes clear that using the exact same order-finding techniques as in chapter 2 we can find an approximation to each component of the derivatives of the function in the state

$$= \left| \frac{N}{m} \frac{\bar{\partial f}}{\partial x_1} \right\rangle \left| \frac{N}{m} \frac{\bar{\partial f}}{\partial x_2} \right\rangle \dots \left| \frac{N}{m} \frac{\bar{\partial f}}{\partial x_d} \right\rangle \tag{G.11}$$

The final step is to take a measurement in the computational basis to obtain the components of ∇f with n bits of precision. On a classical computer approximating the numerical gradient of a function requires a minimum of $d+1$ black-box queries, whereas on a quantum computer it requires only a single query regardless of d where d is the dimension of the function. Further analysis can be found in Stephen Jordan's paper [95].

Appendix H

Quantum heuristics: Quantum particle swarm optimization algorithm

H.0.1 Introduction to quantum algorithmic particle swarm optimization

Qubit representation and Operations

Although the Grover algorithm is currently the best algorithm for solving heuristic problems, propositions have been made for quantum versions of heuristic algorithms like the particle swarm. The reason for interest here is the fact that some of these algorithms actually present a speed-up over their classical counterparts. So in this section we will look at a quantum algorithmic version of the particle swarm optimization.

Before going into the quantum version of the particle swarm optimization algorithms proposed in [97] we will look at some mathematical properties of a quantum particle beginning with their representation as a qubit as shown below

$$|\phi\rangle = \cos\theta\,|0\rangle + \sin\theta\,|1\rangle = [\cos\theta\ \sin\theta]^T. \tag{H.1}$$

Now we can perform transformations on our quantum particle with operations such as $\mathbf{R}(\boldsymbol{\Delta}\theta)$ given below

$$\mathbf{R}(\Delta\theta) = \begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & \cos\Delta\theta \end{bmatrix}. \tag{H.2}$$

Taking

$$|\phi\rangle = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}. \tag{H.3}$$

it can be transformed by $\mathbf{R}$ as shown

$$\mathbf{R}\,|\phi\rangle = \begin{bmatrix} \cos(\theta+\Delta\theta) \\ \sin(\theta+\Delta\theta) \end{bmatrix}. \tag{H.4}$$

Letting

$$|\phi_i\rangle = \cos\theta_\mathrm{i}\,|0\rangle + \sin\theta_\mathrm{i}\,|1\rangle\,. \tag{H.5}$$

it follows that $|\phi_1\phi_2\ldots\phi_n\rangle$ can be written as

$$|\phi_1\phi_2\ldots\phi_n\rangle = |\phi_1\rangle\otimes|\phi_2\rangle\ldots\otimes|\phi_n\rangle\,. \tag{H.6}$$

which can be written in vector form as

$$\left[\begin{array}{c} \cos\theta_1 \\ \sin\theta_1 \end{array}\right] \otimes \ldots \otimes \left[\begin{array}{c} \cos\theta_\mathrm{n} \\ \sin\theta_\mathrm{n} \end{array}\right] = \left[\begin{array}{cccc} \cos\theta_1\times & \cos\theta_2\times & \ldots & \times\cos\theta_\mathrm{n} \\ \cos\theta_1\times & \cos\theta_2\times & \ldots & \times\sin\theta_\mathrm{n} \\ \vdots & \vdots & & \vdots \\ \sin\theta_1\times & \sin\theta_2\times & \ldots & \times\sin\theta_\mathrm{n} \end{array}\right]. \tag{H.7}$$

In an n qubit system of the ground state probability amplitudes is a function of an n qubit phase $(\phi_1, \phi_2, \ldots, \phi_n)$ *i.e* adjustments of ϕ_i update all 2^n probability amplitudes. Therefore the operation $\mathbf{R}(\boldsymbol{\Delta\theta_i})$ also updates the probability amplitudes. The operation $\mathbf{R}(\boldsymbol{\Delta\theta_1\Delta\theta_1\ldots\Delta\theta_n})$ on all qubits can be written more clearly as

$$\mathbf{R}(\boldsymbol{\Delta\theta_1\Delta\theta_1\ldots\Delta\theta_n}) = \mathbf{R}(\boldsymbol{\Delta\theta_1}) \otimes \mathbf{R}(\boldsymbol{\Delta\theta_2}) \otimes \ldots \otimes \mathbf{R}(\boldsymbol{\Delta\theta_n}). \tag{H.8}$$

where

$$\mathbf{R}(\boldsymbol{\Delta\theta_i}) = \left[\begin{array}{cc} \cos(\Delta\theta_\mathrm{i}) & -\sin(\Delta\theta_\mathrm{i}) \\ \sin(\Delta\theta_\mathrm{i}) & \cos(\Delta\theta_\mathrm{i}) \end{array}\right]. \tag{H.9}$$

It follows that when $n = 2$ then

$$\mathbf{R}(\boldsymbol{\Delta\theta_1}, \boldsymbol{\Delta\theta_2}) = \left[\begin{array}{cccc} \cos\Delta\theta_1\cos\Delta\theta_2 & -\cos\Delta\theta_1\sin\Delta\theta_2 & -\sin\Delta\theta_1\cos\Delta\theta_2 & \sin\Delta\theta_1\sin\Delta\theta_2 \\ \cos\Delta\theta_1\sin\Delta\theta_2 & \cos\Delta\theta_1\cos\Delta\theta_2 & -\sin\Delta\theta_1\sin\Delta\theta_2 & \sin\Delta\theta_1\cos\Delta\theta_2 \\ \sin\Delta\theta_1\cos\Delta\theta_2 & -\sin\Delta\theta_1\sin\Delta\theta_2 & -\cos\Delta\theta_1\sin\Delta\theta_2 & \cos\Delta\theta_1\cos\Delta\theta_2 \\ \sin\Delta\theta_1\sin\Delta\theta_2 & \sin\Delta\theta_1\cos\Delta\theta_2 & \cos\Delta\theta_1\sin\Delta\theta_2 & \cos\Delta\theta_1\cos\Delta\theta_2 \end{array}\right]. \tag{H.10}$$

We can write

$$\mathbf{R}(\boldsymbol{\Delta\theta_1\Delta\theta_1\ldots\Delta\theta_n}) \left|\phi_1\phi_2\ldots\phi_n\right\rangle = \left|\hat{\phi_1}\right\rangle \otimes \left|\hat{\phi_1}\right\rangle \otimes \ldots \otimes \left|\hat{\phi_n}\right\rangle. \tag{H.11}$$

where $\left|\hat{\phi_i}\right\rangle = \cos(\theta_i + \Delta\theta_i)\left|0\right\rangle + \sin(\theta_i + \Delta\theta_i)\left|1\right\rangle$

Encoding Method Based on Multiqubits Phases

Since we now know how to perform operation on qubit states we are interested now in formulating a method for encoding the particle information. We begin by randomly generating N (number of particles), n dimensional phase vectors θ_i where $i = 1, 2, \ldots, N$ and

$$\theta_{ij} = [\theta_{i1}, \theta_{i2}, \ldots, \theta_{in}]. \tag{H.12}$$

where $\theta_{ij} = 2\pi r$, for $0 \leq r \leq 1$, where $j = 1, 2, \ldots, n$. Let $|\theta_{ij}\rangle = \cos\theta_{ij}|0\rangle + \sin\theta_{ij}|1\rangle$, we can thus generate the N, n-qubit vectors as $|\theta_{11}, \theta_{12}, \ldots, \theta_{1n}\rangle; |\theta_{21}, \theta_{22}, \ldots, \theta_{2n}\rangle; \ldots; |\theta_{N1}, \theta_{N2}, \ldots, \theta_{Nn}\rangle$.

Phase Updating

Multi-qubit rotation gates are employed to update particles. Let the phase vector of the global optimal particle be $\theta_g = [\theta_{g1}, \theta_{g2}, \ldots, \theta_{gn}]$ and let the phase vectors of the i^{th} particle be $\theta_i = [\theta_{i1}, \theta_{i2}, \ldots, \theta_{in}]$ and the personal best phase vector be $\theta_{bi} = [\theta^i{}_{b1}, \theta^i{}_{b2}, \ldots, \theta^i{}_{bn}]$. It follows that if θ_i is updated then it's corresponding amplitude is also updated. Improving search capability requires that all phases θ_i be updated, which allows all particles to be updated N times. Let $\Delta\theta_0$ denote an initial non-zero step size. The quantum particle swarm algorithm can be given by the following steps,

Step 1: Prepare your state such that the probability amplitudes of the states are given as

$$A_i(\theta) = \begin{bmatrix} \cos\theta_{\mathrm{i1}} \\ \sin\theta_{\mathrm{i1}} \end{bmatrix} \otimes \ldots \otimes \begin{bmatrix} \cos\theta_{\mathrm{in}} \\ \sin\theta_{\mathrm{in}} \end{bmatrix} = \begin{bmatrix} \cos\theta_{\mathrm{i1}} & \cos\theta_{\mathrm{i2}} & \ldots \cos\theta_{\mathrm{i1}} \\ \cos\theta_{\mathrm{i1}} & \cos\theta_{\mathrm{i2}} & \ldots \sin\theta_{\mathrm{i2}} \\ \vdots & \vdots & \vdots \\ \sin\theta_{\mathrm{i1}} & \sin\theta_{\mathrm{i2}} & \ldots \sin\theta_{\mathrm{in}} \end{bmatrix}. \tag{H.13}$$

Step 2: Set the update rotation angles to 0 *i.e* $\Delta\theta_{i1} = \Delta\theta_{i2}$=...= $\Delta\theta_{in}$ =0
Step 3: Determine the value of the rotation angle where *sgn* denotes the symbolic function. If

$$|{\theta^i}_{bj} - \theta_{ij}| \leq \pi \tag{H.14}$$

then

$$\Delta{\theta^i}_{bj} = \mathrm{sgn}({\theta^i}_{bj} - \theta_{ij})\Delta\theta_0 \tag{H.15}$$

If

$$|{\theta^i}_{bj} - \theta_{ij}| \geq \pi \tag{H.16}$$

then

$$\Delta{\theta^i}_{bj} = -\mathrm{sgn}({\theta^i}_{bj} - \theta_{ij})\Delta\theta_0 \tag{H.17}$$

If

$$|{\theta^i}_{gj} - \theta_{ij}| \geq \pi \tag{H.18}$$

then

$$\Delta{\theta^i}_{gj} = -\mathrm{sgn}({\theta^i}_{gj} - \theta_{ij})\Delta\theta_0 \tag{H.19}$$

If

$$|{\theta^i}_{gj} - \theta_{ij}| \leq \pi \tag{H.20}$$

then

$$\Delta{\theta^i}_{gj} = \mathrm{sgn}({\theta^i}_{gj} - \theta_{ij})\Delta\theta_0 \tag{H.21}$$

Step 4: Calculate the rotation angles and update the rotation angle using the equation

$$\Delta\theta_{ij} = r1 \times \Delta{\theta^b}_{ij} + r2 \times \Delta{\theta^g}_{ij} \tag{H.22}$$

where $r1$ and $r2$ are random numbers, then

$$A_{i+1}(\theta) = R_n(\Delta\theta_{i1}, \Delta\theta_{i2}, \ldots, \Delta\theta_{in})A_i(\theta) \tag{H.23}$$

Step 5: If $j < n$ then take another iteration $j + 1$ *i.e* return to step 3

Bibliography

[1] M. Nielsen, I. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge and New York, 2000.

[2] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing 26 (1997) 1484–1509.

[3] Lov K. Grover, Quantum mechanics helps in searching for a needle in a haystack. Physical Review Letters, 79(2):325-328, 1997. arXiv:quant-ph/9605043.

[4] S. Jordan, Quantum algorithm zoo, `http://math.nist.gov/quantum/zoo/` (accessed February 16, 2015).

[5] R. Feynman, Simulating physics with computers, International Journal of Theoretical Physics 21 (1982) 467–488.

[6] D. Deutsch, Quantum theory, the church-turing principle and the universal quantum computer, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 400 (1818) (1985) 97–117.

[7] Church, A. Abstract No. 204. Bull. Amer. Math. Soc. 41, 332-333, 1935.

[8] A. Berthiaume, G. Brassard, Oracle quantum computing, J. Modern Opt. 41 (1994) 25212535.

[9] D. Boneh, J. Lipton, Quantum cryptanalysis of hidden linear functions, in: D. Coppersmith (Ed.), Crypto '95, Lecture Notes in Computer Science, Springer Verlag, 1995, pp. 424–437.

[10] P. Shor ,Algorithms for Quantum Computation: Discrete Logarithms and Factoring.

[11] R. Jozsa, L. N., On the role of entanglement in quantum computational speed-up, Proc. Roy. Soc. London A 459 (2003) 2011–2032.

[12] M. Boyer, G. Brassard, P. Hyer, and A. Tapp ,Tight bounds on quantum searching. Fortschritte der Physik, 46:493-505, 1998.

[13] C. Drr, P. Hyer, A quantum algorithm for finding the minimum. arXiv:quant-ph/9607014, 1996.

[14] Ashwin. Nayak, Felix. Wu, The quantum query complexity of approximating the median and related statistics. In Proceedings of 31st ACM Symposium on the Theory of Computing, 1999. arXiv:quant-ph/9804066.

[15] L. A. B. Kowada, C. Lavor, R. Portugal, C. M. H. de Figueiredo ,A new quantum algorithm for solving the minimum searching problem International Journal of Quantum Information, Vol. 6, No. 3, pg. 427-436, 2008.

[16] G. Brassard, P. Hyer, and A. Tapp ,Quantum counting. arXiv:quant-ph/9805082, 1998.

[17] M. Mosca , Quantum searching, counting, and amplitude amplification by eigenvector analysis. In R. Freivalds, editor, Proceedings of International Workshop on Randomized Algorithms, pages 90-100, 1998.

[18] Erich. Novak, Quantum complexity of integration. Journal of Complexity, 17:2-16, 2001. arXiv:quant-ph/0008124.

[19] C Drr,M Heiligman,P Hyer, M Mhalla, Quantum query complexity of some graph problems. SIAM Journal on Computing, 35(6):1310-1328, 2006.arXiv:quant-ph/0401091.

[20] L. Grover, Fixed-point quantum search. Phys. Rev. Lett. 95(15):150501, 2005. arXiv:quant-ph/0503205

[21] T. Tulsi, L. Grover, A. Patel, A new algorithm for fixed point quantum search. Quantum Information and Computation 6(6):483-494, 2005. arXiv:quant-ph/0505007

[22] E. Biham, O. Biham, D. Biron, M. Grassl, and D. Lidar , Grover's quantum search algorithm for an arbitrary initial amplitude distribution. Physical Review A, 60(4):2742, 1999. arXiv:quant-ph/9807027 and arXiv:quant-ph/0010077

[23] A. Montanaro, Quantum search with advice. In Proceedings of the 5th conference on Theory of quantum computation, communication, and cryptography (TQC 2010) arXiv:0908.3066

[24] G. Brassard, P. Hyer, M. Mosca, A. Tapp, Quantum amplitude amplification and estimation. In Samuel J. Lomonaco Jr. and Howard E. Brandt, editors, Quantum Computation and Quantum Information: A Millennium Volume, volume 305 of AMS Contemporary Mathematics Series. American Mathematical Society, 2002. arXiv:quant-ph/0005055

[25] A. Ambainis, Quantum Search Algorithms. SIGACT News, 35 (2):22-35, 2004. arXiv:quant-ph/0504012

[26] N.J. Cerf, L.K. Grover, C.P. Williams, Nested quantum search and NP-hard problems. Applicable Algebra in Engineering, Communication and Computing, 10 (4-5):311-338, 2000.

[27] D. Cornwell, Amplified Quantum Transforms arXiv:1406.0190, 2015.

[28] T. Laarhoven, M. Mosca, and J. van de Pol, Solving the shortest vector problem in lattices faster using quantum search Proceedings of PQCrypto13, pp. 83-101, 2013. arXiv:1301.6176

[29] A. Childs, J. Goldstone, Spatial search by quantum walk Physical Review A, 70:022314, 2004. arXiv:quant-ph/0306054

[30] S. Chakraborty, L. Novo, A. Ambainis, Y. Omar, Spatial search by quantum walk is optimal for almost all graphs arXiv:1508.01327, 2015.

[31] T.G. Wong, Quantum walk search on Johnson graphs arXiv:1601.04212, 2016.

[32] P. Dimitri , A. Nedic, A. Ozdaglar (2003), Convex Analysis and Optimization. Belmont,MA.: Athena Scientific. ISBN 1-886529-45-0.

[33] A. Baha (2012), "Stochastic second-order cone programming: Application models". Applied Mathematical Modelling. 36 (10): 51225134. doi:10.1016/j.apm.2011.12.053.

[34] R. Courant, D. Hilbert(1953), Methods of Mathematical Physics. Vol. I (First English ed.). New York: Interscience Publishers, Inc. p. 169. ISBN 978-0471504474.

[35] P. Judea (1984), Heuristics: intelligent search strategies for computer problem solving. United States: Addison-Wesley Pub. Co., Inc., Reading, MA. p. 3.

[36] R. Duffin, E. Peterson, C. Zener (1967), Geometric Programming. John Wiley and Sons. p. 278. ISBN 0-471-22370-0.

[37] P. Stephen. Vandenberghe, Lieven (2004). Convex Optimization (pdf). Cambridge University Press. ISBN 978-0-521-83378-3.

[38] P.B. Dimitri (1999), Nonlinear Programing (Second ed.). Cambridge, MA.: Athena Scientific. ISBN 1-886529-00-0.

[39] J.R. Birge, F.V. Louveaux, Introduction to Stochastic Programming. Springer Verlag, New York, 1997.

[40] F. L. Hitchcock , The distribution of a product from several sources to numerous localities, Journal of Mathematics and Physics, 20, 1941, 224-230.

[41] A. Mordecai (2003), Nonlinear Programming: Analysis and Methods. Dover Publishing. ISBN 0-486-43227-0.

[42] J. Janmark, D. A. Meyer, T.G. Wong , Global symmetry is unnecessary for fast quantum search Physical Review Letters 112:210502, 2014. arXiv:1403.2228

[43] D.A. Meyer, T.G. Wong , Connectivity is a poor indicator of fast quantum search Physical Review Letters 114:110503, 2014. arXiv:1409.5876

[44] T.G. Wong, Spatial search by continuous-time quantum walk with multiple marked vertices Quantum Information Processing 15(4):1411-1443, 2016. arXiv:1501.07071

[45] A. Kitaev, A. Shen, A. Vlyayi, Classical and Quantum Computation (Graduate Studies in Mathematics),2002.

[46] C. Gidney, http://twistedoakstudios.com/blog/Post2644 grovers quantum search algorithm,2013.

[47] G. C. Stokes, Trans. Cambr. Phil. Soc. 9, 399 (1852).

[48] E. Hecht, A. Zajac, Optics (Addision-Wesley, Reading MA, 1974) section 8.12.

[49] Daniel F. V. James, Paul G. Kwiat, William J. Munro and Andrew G. White, On the Measurement of Qubits,2001.

[50] A. Miar, A. Virzani, G. Weighs, A. Zeilinger, Entanglement of the orbital angular momentum states of photons, Nature 412,313-316(2001).

[51] E.J. Galvez, C.J Holbrow, M.J Pysher, J.W Martin,N. Courtemanche,L. Heilig,J. Spencer Interference of correlated photons:five quantum mechanics experiments for undergraduates (2004).

[52] H. Bassa, Masters thesis, Implementing Grover's search algorithm using one way quantum computing model and photonic orbital angular momentum.

[53] S Koziel, J.W. Bandler, "Space mapping with multiple coarse models for optimization of microwave components," IEEE Microwave and Wireless Components Letters, vol. 8, no. 1, pp. 13, Jan. 2008.

[54] D.E Bishnu Prasa Kar, R. Mandal, D. Ghoshal, S. P. (2014-09-27), "Optimal selection of components value for analog active filter design using simplex particle swarm optimization". International Journal of Machine Learning and Cybernetics.

[55] N Friedrich, Space mapping outpaces EM optimization in handset-antenna design, microwaves and rf, Aug. 30, 2013.

[56] M.S Lobo, L. Vandenberge, S. Boyd, H. Lebret, Applications of second-order cone programming, Linear Algebra and its Applications,193-228(1998).

[57] L. Galli, Algorithms for Integer Programming,(2014).

[58] www.mathworks.com/discovery/quadratic-programming.html

[59] S. Schaible,J. Shi, Recent developments in fractional programming:single ratio and min-max case.

[60] L. Hannah, Stochastic Optimization.

[61] A. Ben-Tal, L. Ghauoli,A. Nemirovski, Robust Optimization,Princeton University Press Princeton and Oxford,2008.

[62] H. Mohammed,H. Baker,J.W. Blander Introduction to the Space Mapping Technique,2002.

[63] A. Krzysztof (2003), Principles of constraint programming. Cambridge University Press. ISBN 0-521-82583-0.

[64] L.C. Evans, Introduction to Mathematical optimal control theory version 2 notes, University of Berkeley.

[65] R.A. Howard, Dynamic programming and Markov Processes, John Wiley and sons,1960.

[66] J. Cook , H. Cunningham, R. Pulleyblank, A. Schrijver (1997), Combinatorial Optimization. Wiley. ISBN 0-471-55894-X.

[67] Z. Luo ,J.S. Pang ,D. Ralph, Mathematical Programs with Equilibrium constraints,Cambridge University Press,1996.

[68] A. Alfaris, Multidisciplinary System design Optimization ,Multiobjective optimization notes,MIT notes.

[69] I. Zintchenko,E. Brown,M. Troyer, Recent developments in quantum annealing.

[70] N. Wiebe,D. Braun, S. Lloyd, (2 August 2012), "Quantum Algorithm for Data Fitting". Physical Review Letters. 109 (5): 050505. Bibcode:2012PhRvL.109e0505W. PMID 23006156. arXiv:1204.5242Freely accessible. doi:10.1103/PhysRevLett.109.050505

[71] P. Wittek,Quantum Machine Learning, Cambridge University Press, Cambridge and New York, 2014.

[72] B. Dickson, "Exploiting machine learning in cybersecurity". TechCrunch. Retrieved 2017-05-23.

[73] Wernick, Yang, Brankov, Yourganov and Strother, Machine Learning in Medical Imaging, IEEE Signal Processing Magazine, vol. 27, no. 4, July 2010, pp. 2538

[74] S. Jordan, A fast Quantum Algorithm for Numerical Gradient Estimation, 2005.

[75] Edward. Farhi, Jeffrey. Goldstone, Sam. Gutmann, and Michael. Sipser .Invariant quantum algorithms for insertion into an ordered list. arXiv:quant-ph/9901059, 1999.

[76] A. M. Childs, A. J. Landahl, P. A. Parrilo Quantum algorithms for the ordered search problem via semidefinite programming.Physical Review A, 75 032335, 2007. arXiv:quant-ph/0608161

[77] M. Ben-Or , Avinatan , Quantum search in an ordered list via adaptive learning.arXiv:quant-ph/0703231, 2007.

[78] A. Childs , T. Lee , Optimal quantum adversary lower bounds for ordered search. Proceedings of ICALP 2008 arXiv:0708.3396

[79] E. Farhi, J. Goldstone, S. Gutmann , A quantum approximate optimization algorithm arXiv:1411.4028, 2014.

[80] C. Yen-Yu. Lin , Y. Zhu, Performance of QAOA on typical instances of constraint satisfaction problems with bounded degree arXiv:1601.01744, 2016.

[81] D. Wecker, M.B. Hastings, M. Troyer ,Training a quantum optimizer arXiv:1605.05370, 2016.

[82] Z-C Yang, A. Rahmani, A. Shabani, H. Neven, C. Chamon, Optimizing variational quantum algorithms using Pontryagins's minimum principle arXiv:1607.06473, 2016.

[83] C. Gidney, http://twistedoakstudios.com/blog/Post2644 grovers quantum search algorithm,2013.

[84] D. Bulger, Quantum basin hopping with gradient-based local optimisation. arXiv:quant-ph/0507193, 2005.

[85] S.P. Jordan, Quantum Computation Beyond the Circuit Model.PhD thesis, Massachusetts Institute of Technology, 2008. arXiv:0809.2307.

[86] A. Yao, On computing the minima of quadratic forms.In Proceedings of the 7th ACM Symposium on Theory of Computing, pages 23-26, 1975.

[87] T. Hogg ,Highly structured searches with quantum computers.Physical Review Letters 80: 2473, 1998.

[88] M. Hunziker, D.A. Meyer, Quantum algorithms for highly structured search problems. Quantum Information Processing, Vol. 1, No. 3, pg. 321-341, 2002.

[89] D.A. Meyer, J. Pommersheim, Single-query learning from abelian and non-abelian Hamming distance oracles arXiv:0912.0583.

[90] M. Rtteler,Quantum algorithms to solve the hidden shift problem for quadratics and for functions of large Gowers norm. In Proceedings of MFCS 2009, pg 663-674.arXiv:0911.4724.

[91] T. Hogg, Highly structured searches with quantum computers.Physical Review Letters 80: 2473, 1998.

[92] www.quantumzoo.com ,2013, maintained by Stephen Jordan.

[93] F.G.S.L. Brandao and K. Svore, Quantum speed-ups for semidefinite programming arXiv:1609.05537, 2016.

[94] D. Poulin, P. Wocjan, Sampling from the thermal quantum Gibbs state and evaluating partition functions with a quantum computer. Physical Review Letters 103:220502, 2009. arXiv:0905.2199

[95] S. Jordan,A fast Quantum Algorithm for Numerical Gradient Estimation, 2005.

[96] E. Bernstein, U. Vazirani,1993), "Quantum complexity theory", Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC '93), pp. 1120, doi:10.1145/167088.16709

[97] X. Li, H. Xu, X. Guan, Quantum-Inspired Particle Swarm Optimization Algorithm Encoded by Probability Amplitudes of Multi-Qubits, Open Journal of Optimization, 2015, 4, 21-30.Published Online June 2015 in SciRes. http://www.scirp.org/journal/ojop http://dx.doi.org/10.4236/ojop.2015.42003 2015.

[98] P. Rebentrost,M. Schuld,F. Petruccione,S. Lloyd ,Quantum gradient descent and Newton's method for constrained polynomial optimization, 2016.

[99] E. Bernstein,V. Vazirani, Quantum complexity theory In proceedings of the 25th ACM Symposium on the theory of computing,pages 11-20, 1993.

[100] T. Kadowaki,H. Nishimori, "Quantum annealing in the transverse Ising model" Phys. Rev. E 58, 5355 (1998)

[101] V. Giovannetti,S. Lloyd, L. Maccone Quantum random access memory, 2007.

[102] C. Cade, A. Montanaro, A. Belovs ,Time and space efficient quantum algorithms for detecting cycles and testing bipartiteness arXiv:1610.00581, 2016.

[103] G. Wang, Span-program-based quantum algorithm for tree detection.arXiv:1309.7713, 2013.

[104] T. Carette, M. Laurire, F. Magniez,Extended learning graphs for triangle finding arXiv:1609.07786, 2016.

[105] E. Farhi, J. Goldstone, S. Gutmann, M. Sipser ,Quantum computation by adiabatic evolution.arXiv:quant-ph/0001106, 2000.

[106] L.E. Ohanian, E.C. Prescott, N.L. Stokey, Introduction to dynamic general equilibrium. Journal of Economic Theory 144 (2009) 22352246, 2009

[107] I. Kassal, S. Jordan, P. Love, M. Mohseni, A. Aspuru-Guzik, Quantum algorithms for the simulation of chemical dynamics.Proc. Natl. Acad. Sci. Vol. 105, pg. 18681, 2008. arXiv:0801.2986.

[108] D. Abrams ,S. Lloyd ,Simulation of many-body Fermi systems on a universal quantum computer. Physical Review Letters, 79(13):2586-2589, 1997. arXiv:quant-ph/9703054

[109] T. Byrnes,Y. Yamamoto, Simulating lattice gauge theories on a quantum computer.Physical Review A, 73, 022328, 2006. arXiv:quant-ph/0510027.

[110] A. Harrow, A. Hassidim, S. Lloyd ,Quantum algorithm for solving linear systems of equations. Physical Review Letters 15(103):150502, 2009.arXiv:0811.3171.

[111] S. Lloyd,S. Garnerone, P. Zanardi ,Quantum algorithms for topological and geometric analysis of big data arXiv:1408.3106

[112] E. Farhi, J. Goldstone, S. Gutmann, M. Sipser, Quantum computation by adiabatic evolution.arXiv:quant-ph/0001106, 2000.

[113] S. Jansen, M. Ruskai, R. Seiler , Bounds for the adiabatic approximation with applications to quantum computation. Journal of Mathematical Physics, 48:102111, 2007.arXiv:quant-ph/0603175.

[114] A. Ethem (2010), Introduction to Machine Learning. MIT Press. pg9. ISBN 978-0-262-01243-0.

[115] E.G. Steward (2008), Quantum Mechanics: Its Early Development and the Road to Entanglement. Imperial College Press. ISBN 978-1-86094-978-4.

[116] A. Sen, M. Srivastava, Regression Analysis Theory, Methods, and Applications, Springer-Verlag, Berlin, 2011 (4th printing).

[117] B. Ken (1994),"Numerical Taxonomy and Cluster Analysis". Typologies and Taxonomies. p. 34. ISBN 9780803952591.

[118] D. Luenberger(1997),Optimization by Vector Space Methods.John Wiley and Sons.ISBN 0-471-18117-X.

[119] T. Roweis, K. Saul(2000), "Nonlinear Dimensionality Reduction by Locally Linear Embedding". Science. 290 (5500): 23232326. PMID 11125150. doi:10.1126/science.290.5500.2323.

[120] F. Bonnans, C. Gilbert, C. Lemarchal; A. Sagastizbal(2006), Numerical optimization: Theoretical and practical aspects. Universitext (Second revised ed. of translation of 1997 French ed.). Berlin: Springer-Verlag. pp. xiv+490. ISBN 3-540-35445-X. MR 2265882. doi:10.1007/978-3-540-35447-5.

[121] B. Dantzig , N.Thapa(2003), Linear Programming 2: Theory and Extensions. Springer-Verlag.

[122] J. Snyman (3 March 2005), Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms. Springer Science and Business Media. ISBN 978-0-387-24348-1.

[123] M. Grtschel, L. Lovsz, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer, 1988.

[124] S. Bhatnagar , L .Prasad , and A . Prashanth (2013), Stochastic Recursive Algorithms for Optimization: Simultaneous Perturbation Methods, Springer

[125] J. Kennedy,R. Eberhart (1995),Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks. IV. pp. 19421948. doi:10.1109/ICNN.1995.48896

[126] E. Rashedi,.H. Nezamabadi-pour, A Gravitational Search Algorithm,Science Direct,Information Sciences.Volume 179, Issue 13, 13 June 2009, Pages 2232-2248 (2009)

[127] I. Bengtsson , K. yczkowski (2006), Geometry of Quantum States. An Introduction to Quantum Entanglement. Cambridge: Cambridge University Press. second, revised edition (2017)

[128] C. Pomerance, A Tale of Two Sieves. Not. Amer. Math. Soc. 43, 1473-1485, 1996.

[129] D. Coppersmith, Modifications to the Number Field Sieve. J. Cryptology 6, 169-180, 1993.

[130] H. Uys, H. Bassa, P.J.W du Toit, S. Gosh, T. Konrad, Quantum Control through a Self-Fulfilling Prophecy, arXiv:1705.10356v1 [quant-ph],2017.

[131] K. Dervis, An Idea Based on Honey Bee Swarm For Numerical Optimization.Technical report.Erciyes University, Engineering Faculty Computer Engineering Department Kayseri/Trkiye(2005)