Study on Discrete Quantum Adiabatic Computation in Combinatorial Search problems

March 2012

Mohamed El-Fiky

Contents

List of Figures

List of Tables

Acknowledgments

First of all, I would like to extend my deepest gratitude to my supervisor, Professor Dr. Shigeru Nakayama and to associate Professor Dr. Satoshi Ono for all useful advices, comments and discussions about my research. During my study at the Graduate School of Science and Engineering, Kagoshima University, they have been my invaluable human resources.

Many thanks for all of my colleagues at Media Engineering Laboratory for their valuable academic discussions and effective supports.

I am indebted to my family and friends who always wished for me the best and for love and support.

Dedication

I dedicate this thesis to my family

Abstract

The model of adiabatic quantum computation is a new paradigm for designing quantum algorithms. This model is based on the quantum adiabatic theorem, where a quantum computer evolves the quantum system slowly to switch gradually from an initial Hamiltonian with a ground state which is easy to construct, to a final Hamiltonian whose ground state encodes the solution of the problem to be solved.

This dissertation presents an experimental study on the discrete quantum adiabatic computation in combinatorial search problems, we take k-Satisfiability problem as target problem to be examined. For $k \geq 3$, k-SAT is NP-complete, i.e., among the most difficult NP problems in the worst case. Here we proposes a new monotonic variation method for the phase shift and mixing functions in the adiabatic quantum algorithm called the Quadric variation method, in order to speed up the algorithm search and decrease the overall resulting search cost. In addition, we present a better parameter configuration for the algorithm to be used with quadric variation as well as the previously proposed methods

The experiment are carried out in solving random instances of 3-SAT problems. The results indicate that the proposed method of using monotonic quadric variation in the phase functions of the adiabatic algorithm greatly improve the search behavior of the adiabatic algorithm and reduces the resulting search cost, also the results indicate that the better configuration of the algorithm parameters could greatly enhance the search behavior for the proposed method as well as the previously proposed methods.

Chapter 1

Introduction

1.1 Historical Background

The need to do the computation faster and more reliability was and still an important aim. The Scientists and researchers trying to find ways to achieve this goal by designing devices/machines to accelerate the process of the computation, starting from using their own hands passing by very simple devices like, for example, the Abacus [1], ending with the conventional computers we know nowadays .and the development and research efforts are still and will always trying to achieve this goal. The conventional computers we know nowadays (based on Turning Machines) represents the culmination of years of technological advancements beginning with the early ideas of Charles Babbage (1791-1871). The real start was after Johan Barden, Walter Brittan, and Will Shockley developed the transistor in 1947. For inventing new families of computers with more powerful capabilities, it was necessary to increase the number of the transistors used on approximately the same physical space. The only way to achieve this goal is by decreasing the size of the components with a process known as Miniaturization [2]. According to Moor's law: The number of transistors per square inch on integrated circuits has doubled every year since the integrated circuit was invented. This law has been proved to be true since 1965 when Gordon more Moore proposed it until now. However, according to this law, the size of the components will hit the quantum level by 2020 where it is not possible any more to control the states of the components by conventional means. So the need to find alternatives rose up, and the efforts started for that, whereas implementing

quantum computer was one of the strong alternatives [3].

There have been efforts started trying to find suitable alternatives to the conventional computers. Charles Bennet in the early 1970 showed that [4] the computation could be done in a reversible way (where the input can be generated from the output) by describing a Universal Reversible Turning Machine. Edward Fredkin and Tomas Toffoli examined how reversible computation could be done using traditional Boolean logic gates. They have showed that there exists a universal reversible three bit gate for reversible computation known as Toffoli gate that could be used instead of the Traditional Boolean logic gates like AND,OR, etc. The field of quantum computation starts in the early 1980's with the proposal of Paul Benioff [4] for computers working according to the principles of quantum mechanics, and Richard Feynman [1], who noticed that certain quantum mechanical effects cannot be simulated efficiently on conventional computers, so he suggested that computers working according to the principles of quantum mechanics might perform more efficiently. In 1985, David Deutsch introduced the Universal Quantum Turing machine [5], and the next question was how to use this new device. A new few quantum algorithms were introduced since then by David Deutsch, Richard Josa [6]. In 1994, the field started to arise as one of the hot area of research when Peter Shor [7] surprised the world by describing a polynomial-time quantum algorithm for factorizing integers using the phase estimation technique. In 1996, Lov Grover [8] described another quantum algorithm for searching an unstructured list with quadric speed up over conventional algorithms by using the amplitude amplification technique. Most of the algorithms introduced since then have been based mainly on these two techniques (phase estimation and amplitude amplification). Since then, Quantum computers which can exploit quantum mechanical principles like interference and working on superposition of all possible states simultaneously in parallelism to do computation faster arose as a strong possible alternatives to conventional computers. As a result, the race started among mathematicians, computer scientists, and physicists trying to find new quantum algorithms, quantum complexity theory, and quantum theory of information [3], beside the big challenge to build real quantum computers [13].

Recently a new kind of quantum algorithms have been designed using the adi-

abatic theorem, where the quantum computer evolves slowly for sufficient time T to switch gradually from an initial Hamiltonian with known ground state, to a final Hamiltonian whose ground state encodes the known solution. This concept of quantum computation known as quantum adiabatic evolution was pioneered by E. Farhi et al. [24]. The adiabatic quantum algorithm have been applied to solve various optimization problems, such as satisfiability problems [25] [26], finding cliques in random graphs [27], and set partitioning problem [28], where it has shown to give a polynomial search cost growth on average. Later, the adiabatic version of various quantum algorithm were presented such as adiabatic Grover search algorithm [29], and adiabatic Deutsch-Joza algorithm [31]. Also it was established that the adiabatic model is polynomially equivalent to the stander model of quantum circuits [32].

1.2 Research Objective

The model of adiabatic quantum computation is a new paradigm for designing quantum algorithms, proposed by Farhi et al. [24]. It was recently established that this model is polynomially equivalent to the standard model of quantum circuits [36]. Nevertheless, this model provides a completely different way of constructing quantum algorithms and reasoning about them. Therefore, it is seen as a promising approach for the discovery of substantially new quantum algorithms. The adiabatic quantum algorithm have been applied to solve various optimization problems, such as satisfiability problems [25] [26], finding cliques in random graphs [27], and set partitioning problem [28], where it has shown a promising results. Here we presents an experimental study on the discrete adiabatic algorithm in solving combinatorial search problem. The aim of the research is to speed up the search of the adiabatic quantum algorithm ,i.e., presenting a better variation method (scheduling) for the phase function of the algorithm to decrease the steps required to find the solution without breaking the adiabatically conditions. Also it aims to find a better configuration for the algorithm parameters in order to decrease the overall the resulting search cost as well as improve the stability of the adiabatic algorithm.

1.3 Dissertation Overview

The thesis divided into 6 chapters, Starts with chapter 1 as an introduction explains briefly the historical development for the ideas to find alternatives to conventional computation and how it leads to think about the quantum computation as a strongly possible alternative and the adiabatic evolution as an idea for computation. Also this chapter describe the motivation of research and specify the dissertation structure.

Chapter 2 gives an overview of the basic concepts and ideas related to this research namely the basic principles for the quantum computation such as the qubits, interference, and parallelism. At the same time, it defines the targeted combinatorial search problem, Satisfiability problem. this chapter also describes the conceptual differences between the adiabatic evolution model we study and the original quantum computation model known as circuit model

Chapter 3 describes the novel idea behind the adiabatic quantum evolution, and how the adiabatic theorem could be used to evolve the computer system from the initial state to the target final state. This chapter also presents a numerical example describes how to apply the adiabatic evolution method. Then it proceeds to describe the conceptual differences between the local and the global adiabatic evolution, the definition and the importance of the minimum energy gaps, and how to design a practical adiabatic quantum algorithm.

Chapter 4 reports about the results of the experiments carried out to solve instances of knapsack problems with quantum heuristic search algorithm (QHS). This chapter describes in details the operators and the parameters used in the QHS algorithm. Then it reports on the results obtained from solving random instances for three different types of the knapsack problems, finally it presents a comparison between the search behavior of the QHS and the Genetic algorithm using two types of the constrain handling methods namely Penalty Function and Random Repair. Ends with findings and conclusions

Chapter 5 describes a new variation method for the phase functions of the adiabatic quantum algorithm named Quadric variation method. This method was the first proposed version as partially monotonic variation method. A number of

previously presented variation methods are briefly referenced showing the parameter configuration for each and the corresponding minimum energy gaps. Then it proceeds to outline the main functions available in the simulator we uses for the experiments. it reports on results obtained from the experimental study set up to compare the proposed method with most recently proposed variation methods such as cubic variation and linear variation. The experiments used random instances of 3-SAT problems with difference number of variables up to 20 bits. The chapter concludes with summary of findings and suggestions for future works where the experiments revealed that the quadric variation method improve on the other methods such as linear and cubic in the resulting search cost and shows stable search behavior, However more modification and better settings for the parameter are expected.

Chapter 6 extends the presented work and describes the second version of the quadric variation method as a complete monotonic variation method. In addition, it describes the improved formulas we suggests to find a values for the parameter Δ of the algorithm in order to improve the algorithm search behavior and decrease the resulting search cost. Then it proceeds to present a discussion for the experimental results of solving 3-SAT problems up to 28 bits, and presents a comparison for the resulting probability of finding solution, corresponding minimum energy gaps, and the search costs for all the methods. This chapter ends by concluding the findings and the results reflecting the experiences gathered along the research process.

Chapter 2

Background and Related works

2.1 Basic Principals

In this chapter we will introduce a review for the basic concepts of the quantum computations as Qubit and its main difference to the classical bit, the various representation of the qubit, the quantum measurement, the parallelism, interference, and the entangled states. In addition, we will show the difference between the new paradigm of quantum computation with adiabatic evolution technique and the slandered model of the quantum computation using quantum circuit model.

2.1.1 Quantum Computer

Quantum Computer is a computational device, which can do the computation by exploiting the quantum mechanical principles as parallelism, interference, entanglement, and can operate simultaneously on superpositions of all classical search states, allowing them to evaluate properties of all states in about the same time a classical machine requires for a single evaluation. This property is known as quantum parallelism. Superpositions are described by a state vector, consisting of complex numbers, called probability amplitudes, associated with each classical state. The quantum computation raised as hot area of research after existence of many quantum algorithms could solve problems believed to be intractable. Notable among them are Shore polynomial algorithm for factorings integers [21] and Grover algorithm with quadric speed up in the search over best know classical algorithm by using amplitude amplification [23].

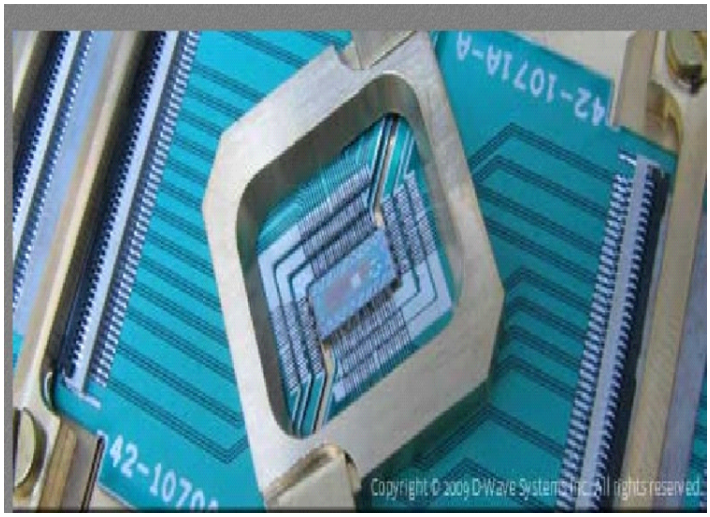


Figure 2.1: First Quantum CPU with 128 qubits, designed and presented by D-Wave systems, February 2007. [12]

Fig. 2.1 presents a first Quantum CPU with 128 qubits, designed and presented by D-Wave systems, February 2007 [12]. Later; this CPU unit was used to present a first real Quantum computer presented by D-wave systems in 2011.

2.1.2 Quantum bit and Superpositions

A classical bit is the basic unit of the information in conventional computers. It is always understood that the bit could be either a 0 or a 1, it can only contain one value at time. A quantum bit or qubit is a basic unit of the information in quantum computation (mechanics). The Qubit also has two possible states $|0\rangle$ and $|1\rangle$ which can be considered as 0, and 1 , respectively in the classical bit. However, the main difference between classical bit and the quantum bit is that the qubit can be exit in linear combination of states $|0\rangle$ and $|1\rangle$ at the same time [4]. this linear combination of possible classical states is called the superposition, and also other important difference is the construction, where the classical bit can be constructed form complete digital circuit called flip-flop, whereas, the qubit can be constructed from a single atom or single molecule [12].

Fig. 2.2 presents an example to show the difference using 3 classical bits and the 3 qubits, 3 classical bit register can carry only one value of the all possible

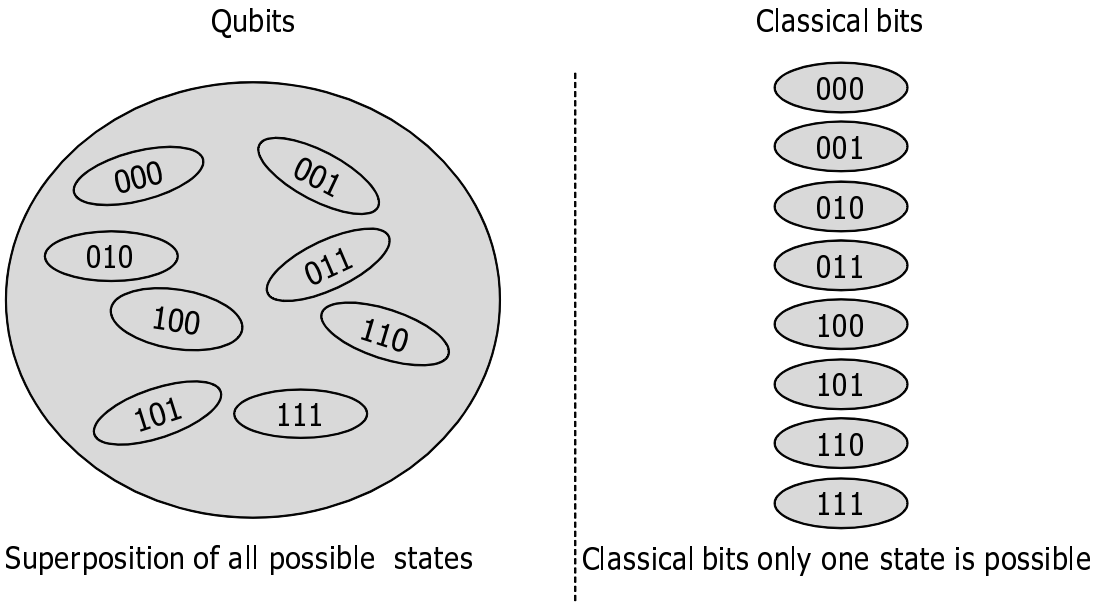


Figure 2.2: The difference between classical bits and qubits in superpositions for example with 3 bits(qubits) [13]

combination for example 001, However, the 3 qubit quantum register can carry one combination as the classical bit or all possible combination in superposition, which gives insights into the power of quantum computation using the parallelism as we well show later in this chapter.

2.1.3 Quantum bit Representations

A qubit in the superposition can be represented as linear combination of states $|0\rangle$ and $|1\rangle$ as follows

$$|\psi\rangle = a|0\rangle + b|1\rangle \tag{2.1}$$

$|\rangle$ notation is called Dirac notation or bra kit notation which it is the standard notation of states in quantum mechanics [11], a and b are complex numbers called the probability amplitudes of the system, and it must be always normalized, i.e., satisfies the condition $|a|^2 + |b|^2 = 1$ where $a = x + iy$, $|a| = \sqrt{aa^*}$, and $a^* = x - iy$ where a^* is the complex conjugate of a. Reading data stored in this qubit while it is in the superposition will break the superposition and output only one state, giving

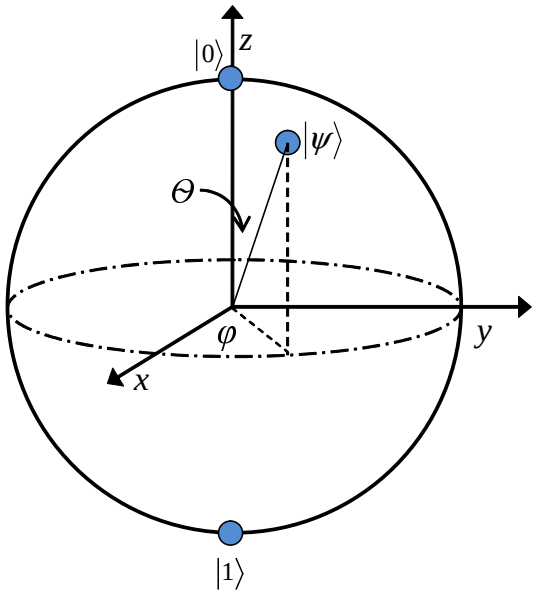


Figure 2.3: A Qubit, a fundamental building block of quantum computers, can be represented as point on The Bloch sphere [39].

either state 0 with probability $|a|^2$ or state 1 with probability $|b|^2$.

For example, a qubit in a perfect superposition could be in the state,

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2.2}$$

Reading (Measuring) this qubit will lead to state $|0\rangle$ with near 50% or state $|0\rangle$ with probability 50%.

Geometric representation of the qubit

One of the important representation of the qubit is the geometric representation, where it shows the effect of the quantum operators in the rotation and phase shift for probability amplitudes of the qubit. The probability amplitudes a, b associated with the states 0 and 1, respectively, are complex number must satisfies the condition $|a|^2 + |b|^2 = 1$ as mentioned before in equation, so we can represent the quantum state of one qubit $|\psi\rangle$ as,

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{2.3}$$

where θ and, ϕ are real numbers defining unique point on the surface of Bloch Sphere shown in the Fig. 2.3 Applying any operation on the qubit will change the values θ and, ϕ to new values consistent with the operation, i.e. rotation [3].

State vector representation of the qubit

The n-qubit quantum system can be represented as vector of length 2^n over Hilbert space. Also it could be presented with the Dirac notation where, it is more useful for describing the quantum state and the evolution of the state to the system . The state of a single qubit system can be represented as a vector in two dimensional complex vector space spanned by the orthogonal basis $|0\rangle$ and $|1\rangle$ as follows,

$$|\psi\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix} a \\ b \end{bmatrix}. \tag{2.4}$$

where,

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.5}$$

and,

$\begin{bmatrix} a \\ b \end{bmatrix}$ is called the system state vector.

2.1.4 Parallelism

The parallelism is one of the most powerful characteristic of the quantum computations. It means the ability of the quantum computer to operate on superpositions of all classical search states simultaneously, i.e., evaluates properties of all possible states in about the same time a classical computer requires for a single evaluation, where any unitary operation could be applied on the system while it is in superposition will be applied on all the states on the system simultaneously, which is one of reasons for the possibility of an exponential speed up of the computation in the quantum computer over the classical computers [2][7].

Fig. 2.4 shows an example using 3 bits (qubit) to show the difference between a quantum operator (function) which can evaluate the all states in superposition in

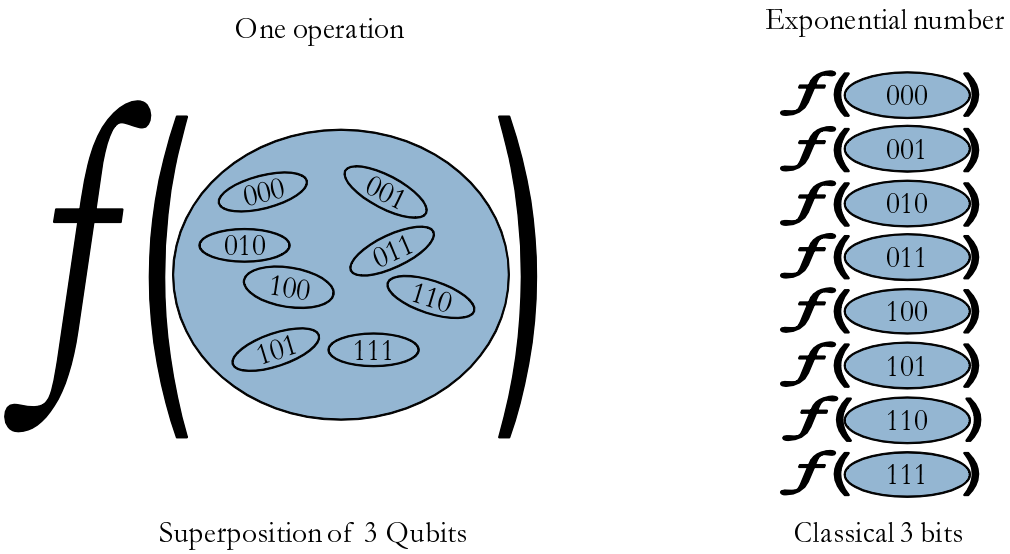


Figure 2.4: A example for Quantum parallelism , i.e., the ability to operate simultaneously on a superposition of all possible classical states. the example uses 3 bits to describe the parallelism [13]

a single evaluation, and the classical computer operator which needs to be applied for each possible state individually ,i.e, 8 times in this case.

2.1.5 Multiple Qubits

Considering a case of 2 bit register, a conventional computer register will be able to carry only one fixed value out of the four possible values 00,01,10,11, at a any time. However, if we consider quantum system (quantum register) with two qubit, the all possible states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ could be found in a superposition. this state in superposition can be represented as,

$$|\psi\rangle = a_0|00\rangle + a_1|01\rangle + a_2|10\rangle + a_3|11\rangle \tag{2.6}$$

where a_i are probability amplitudes (complex numbers) satisfies the condition $\sum_i |a_i|^2 = 1$. Any attempt to read this quantum register will give out one of the four possible states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ with probability $|a_i|^2$ where i is the integer representation of that state.

The tensor product is a mathematical concept important to understand [19], due

to its importance in combine smaller quantum system in a single larger quantum system. For example, let $|\psi\rangle$ and $|\mu\rangle$ be two vectors from a two-dimensional complex vector spanned by basis $|0\rangle$ and $|1\rangle$. The tensor product of $|\psi\rangle$ and $|\mu\rangle$ will be written as,

$$|\psi\rangle \otimes |\mu\rangle \tag{2.7}$$

and have the basis $|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, and |1\rangle \otimes |1\rangle$

Which can shortly written as $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ Similarly, the basis for 3 qubit system will be,

$$|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle \tag{2.8}$$

A two-qubit register in a state vector notation is a complex vector spanned by the orthogonal basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ and can be represented as follows

$$|\psi\rangle = a_0|00\rangle + a_1|01\rangle + a_2|10\rangle + a_3|11\rangle = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}. \tag{2.9}$$

Where

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \tag{2.10}$$

Now, if we consider n-qubit quantum system, this state can be represented as vector of length 2^n over Hilbert space. States can be represented via vector /matrix notation or Dirac notation [11], the Dirac notation is more useful for describing the quantum state and the evolution of the state to the system. And it can be presented as,

$$|\psi\rangle = a_0|00\ldots0\rangle + a_1|00\ldots1\rangle + \ldots + a_n|11\ldots1\rangle \tag{2.11}$$

or shortly

$$|\psi\rangle = \sum_i a_i|i\rangle \tag{2.12}$$

where are probability amplitudes (complex numbers) satisfies the condition $\sum_i |a_i|^2 = 1$ and i is the integer representation of that state .

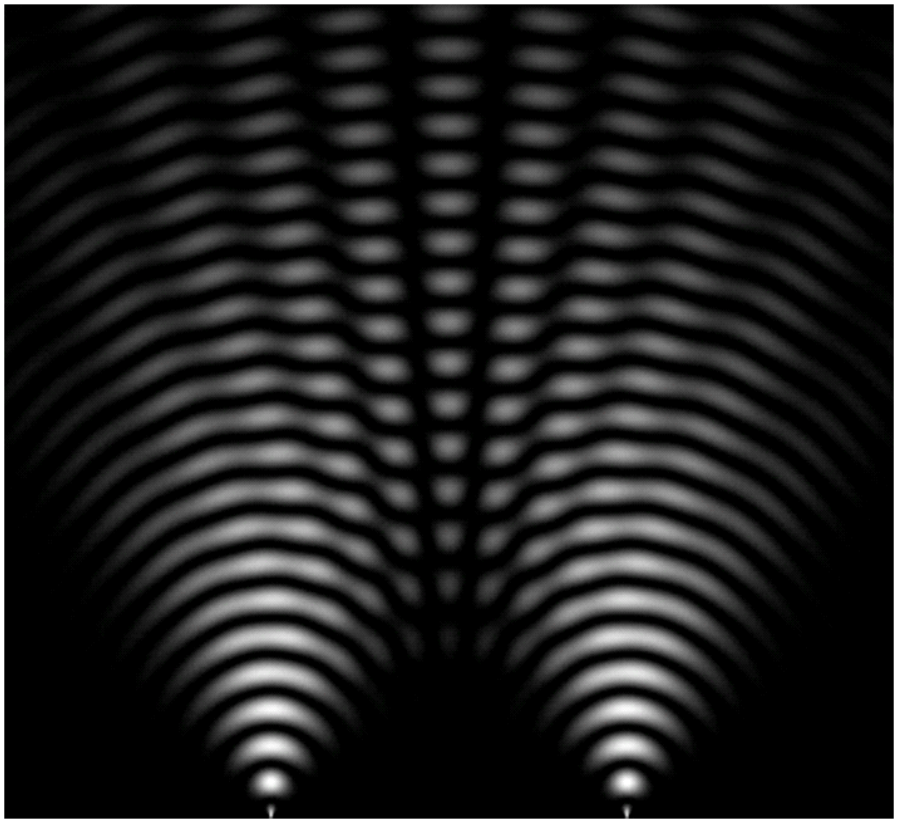


Figure 2.5: The quantum interference phenomena, light waves interfere constructively and destructively [40].

2.1.6 Quantum Interference

Quantum interference is one of the most challenging principles of quantum information theory . Essentially, the concept states that elementary particles can not only be in more than one place at any given time (through superposition), but that an individual particle, such as a photon (light particles) can cross its own trajectory and interfere with the direction of its path. Debate over whether light is essentially particles or waves dates back over three hundred years. In the seventeenth century, Isaac Newton proclaimed that light consisted of a stream of particles; in the early nineteenth century, Thomas Young devised the double-slit experiment to prove that it consisted of waves. Although the implications of Young's experiment are difficult to accept, it has reliably yielded proof of quantum interference through repeated trials. The known physicist Richard Feynman claimed that the essentials of quan-

tum mechanics could be grasped from an exploration of the double slit experiment. For this variation of Young's experiment, a beam of light is aimed at a barrier with two vertical slits. The light passes through the slits and the resulting pattern is recorded on a photographic plate. If one slit is covered, the pattern is what would be expected: a single line of light, aligned with which ever split is open. Intuitively, one would expect that if both slits are open, the pattern of light will reflect that fact: two lines of light, aligned with the slits. In fact, what happens is that the photographic plate is entirely separated into multiple lines of lightness and darkness in varying degrees. What is being illustrated by this result is that interference is taking place between the waves/particles going through the slits, in what, seemingly, should be two non-crossing trajectories.

Fig. 2.5 shows the quantum interference phenomena using two light sources, it shows how the light waves could interfere constructively (wight waves)and destructively (black waves).

The importance and the power of the quantum interference comes from the ability to perform a exponential number of computations in parallel to either cancel or enhance each other [41]. Feynman beautifully describes how light waves can constructively or destructively interfere to produce this effect [42]. The goal of any quantum algorithm is to have a similar phenomena occur, i.e., interference increases the amplitude of computational results we desire and decreases the amplitude of the remaining results. Quantum interference research is being applied in a wide area of applications, such as quantum cryptography, and quantum search algorithms [43].

2.1.7 Entanglement

Entanglement is a non-classical physical phenomenon which has been known since the early days of quantum mechanics [44]. Usually, a state of a quantum system of two or more qubits can be represented in terms of the tensor product of each qubit as we review before, sometimes it is not possible to represent the state of the system in terms of the states of its individual qubits. In such a case, we say that there is a correlation between these components, i.e., each component doesn't have its own state. This is usually referred to as an entangled state [45].

For example, the state $a|00\rangle + b|11\rangle$ cannot be decomposed into the states of two separate qubits, i.e., we cannot find and a_0, a_1, b_0, and b_1 such that,

$$(a_0|0\rangle + b_0|1\rangle) \otimes (a_1|0\rangle + b_1|1\rangle) = a|00\rangle + b|11\rangle \tag{2.13}$$

To satisfy this identity it require that

$$a_0 b_1 = 0 a_1 b_0 = 0 \tag{2.14}$$

However, this implies that a_0 or $b_1$and a_1 or b_0 which is impossible. It means that the state cannot be decomposed to its individual qubits which called entangled state.

Two qubit entangled states are usually known as EPR states, EPR pairs or Bel Basis and can be wrote as following

$$\frac{|00\rangle \pm |11\rangle}{\sqrt{2}}, \frac{|01\rangle \pm |10\rangle}{\sqrt{2}} \tag{2.15}$$

The entangled states are very important and considered as heart for many quantum algorithms, for example quantum teleportation [46], and quantum database search [23].

2.1.8 Quantum Decoherence

Quantum decoherance is a consequence of interaction of quantum systems with their environments resulting in their probabilistic behavior [18]. It is a non-unitary effect (irreversible) and can be viewed as the loss of information from the system to environment. One of the greatest challenges in quantum computation and the practical realization of quantum computers is controlling or removing quantum decoherance. Since they are expected to rely heavily on the undisturbed evolution of quantum coherences. Simply, they require that coherent states be preserved and that decoherance is managed which usually means isolating system from its environment, in order to actually perform quantum computation.

2.1.9 Quantum Measurements

Quantum Measurement is an operation means reading information from any quantum register [19] , measuring a quantum register in a superposition will break that

superposition to some new state that agrees with the outcome of the measurement. For example , consider the two qubit system which can be presented as follow,

$$|\psi\rangle = a_0|00\rangle + a_1|01\rangle + a_2|10\rangle + a_3|11\rangle \tag{2.16}$$

Where a_i is the probability amplitudes (complex numbers) satisfy the condition $\sum_i |a_i|^2 = 1$, if we measure the first qubit to be $|0\rangle$, then the system after-measurement will be in the state,

$$|\psi'\rangle = \frac{1}{\sqrt{|a_0|^2 + |a_1|^2}}(a_0|00\rangle + a_1|01\rangle) \tag{2.17}$$

From Eq. (2.17), we can see that the amplitude of resulting state is still normalized so that its state vector length still equal to 1.

The probability that the first qubit of $|0\rangle$ is given by $|a_0|^2 + |a_1|^2$.

From above we can understand that key of designing any quantum algorithm is that we always need to increase the probability of finding certain state (solution state, matching state) before the measurement operation, so that it could be the output after measurement.

The Measurement can be considered as an operator on the state vector. However, the measurement operators are not unitary, i.e., not reversible unlike all other quantum operators, since any measurement will break the superposition of the state of the system.

2.2 Quantum computation

The field of quantum computation has attracted a great attention in the recent past [20]. This is mainly due to the existence of many algorithms showing that the quantum algorithms based on the principles of quantum mechanics can greatly enhance the efficiency of solving problems believed to be intractable on classical computers. Notable among them are Peter Shor's polynomial time quantum algorithm for factorizing integers [21], the algorithms of Deutsh-Jozsa [22], and Grover algorithm for unstructured search with quadric speed up over any known classical algorithm

[23]. These algorithms follows the slandered model of quantum computation using a quantum logic gates which is known as quantum circuit model.

Recently a novel paradigm to design quantum algorithms using the adiabatic theorem was pioneered by E. Farhi et al. [24], where the quantum computer evolves slowly for sufficient time T to switch gradually from an initial Hamiltonian with known ground state, to a final Hamiltonian whose ground state encodes the known solution. This paradigm of quantum computation is known as quantum adiabatic evolution model.

In Next section we will try to review briefly the basic concepts in this two paradigms.

2.2.1 Quantum circuit model

A Quantum circuit model is the most popular model for quantum computation, in which a computation is a sequence of transformations on n-qubit (quantum) register. These transformations U are unitary, i.e., the inverse of that matrix U^{-1} must be equal to its complex conjugate transpose $U^{\dagger}$ [19], this unitary operators are called quantum gates [5]. The reason that quantum gates must be unitary is that the quantum systems follow the fundamental laws of quantum physics and it must be reversible [6]. Any arbitrary quantum computation (reversible transformation) on any number of qubits can be generated by a finite set of quantum gates (quantum circuit). Such set is said to be universal for quantum computation. An universal gates for classical computation are NAND and NOR gates. In quantum computation, any multiple qubit logic gate can be presented as 2^n x 2^n unitary matrix decomposed from other universal smaller gates.

There are three classes of quantum algorithms which proved to provide an advantage over known classical algorithms.

1. A class of algorithms based upon quantum version of the Fourier transform such as Shor's algorithm for factoring and discrete logarithm [21].

2. Quantum search algorithms for database search known as Grover's algorithm with quadric speed up over best known classical algorithm using amplitude

amplification technique. This technique was the heart for many of Grover like algorithms presented later and proved optimality in many places [23].

3. Quantum simulations (algorithms for simulating quantum systems).

In next section we will review examples for a famous Quantum gates such as Hadamard Gate and phase gate.

Hadamard Gate

Hadamard gate is a pure quantum gate, also it called Hadamard Transform H_n, it can be represented as a $2^n 2^n$ matrix, where n is the number of qubits used. The Hadamard transform can be defined in two ways: recursively, or by using the binary (base-2) representation of the indexes n and k.

1- Recursively, we define the 1 1 Hadamard transform (gate) H_0 by the identity $H_0 = 1$, and then define H_n for $n > 0$ by:

$$H_n = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix} \tag{2.18}$$

Given $H_0 = 1$, then Hadamard transform (gate) for 1 qubit presented as

$$H_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.19}$$

Hadamard gate has special effect when applied to $|0\rangle$ or $|1\rangle$ state qubits, where the output state is the perfect superposition of $|0\rangle$ and $|1\rangle$, i.e., a quantum state that, if observed, will be a 0 or 1 with equal probability. as flowing

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2.20}$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.21}$$

The Hadamard gate (Transform) plays an important role in quantum algorithms where it is used as an initial step, since it could map n qubits initialized with $|0\rangle$ to superposition of all 2^n orthogonal states in the $|0\rangle$ and $|1\rangle$ basis with equal weight [19].

2.2.2 Adiabatic Evolution model

Adiabatic evolution model is a novel paradigm presented by Farhi et al [24] to design new series of quantum algorithms known as adiabatic quantum algorithms, this model of quantum computation relies on the adiabatic theorem, a famous theorem in thermodynamics, to do calculations. The goal is to find a Hamiltonian whose ground state corresponds to the solution of the problem to be solved. First, a system with a simple Hamiltonian (easy to be construct) is taken and initialized to its ground state. Finally, the simple Hamiltonian is adiabatically evolved to the desired Hamiltonian. Then the adiabatic theorem grantee that the system remains in the ground state, under suitable conditions, so that the final state of the system, i.e., after the evolutions for time T will describe the solution to the problem to be solved.

The adiabatic quantum algorithms have been applied to solve various optimization problems, such as satisfiability problems [25] [26], finding cliques in random graphs [27], and set partitioning problem [28], where it has shown to give a polynomial search cost growth on average. Later, the adiabatic version of various quantum algorithm were presented such as adiabatic Grover search algorithm [29] [82], and adiabatic Deutsch-Joza algorithm [31]. Also it was established that the adiabatic model is polynomially equivalent to the slandered model of quantum circuits [36].

2.3 The Quantum Complexly Classes

Computational complexity theory is the subject of classifying the difficulty of various computational problems (both classical and quantum). The basic concept is a complexity class, which can be thought as a collection of computational problems that share some common feature(s) with respect to the computational resources needed to solve those problems. The most important classes are **P** and **NP**. The former is the class of computational problems that can be solved efficiently (in polynomial time) on classical computer and the later is the class of problems which have solutions that can be quickly verified (again in polynomial time) on classical computer [17]. It is clear that **P** is a subset of **NP**, since the ability to solve problem

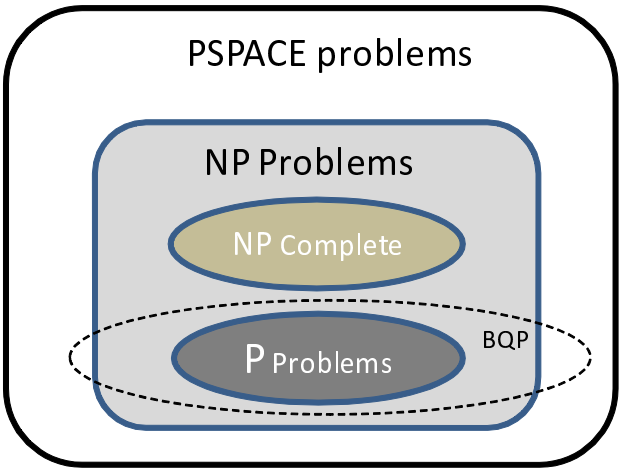


Figure 2.6: The relationship between classical and quantum complexity classes. Where quantum computers fit between P and PSPACE is not known [17].

implies the ability to check potential for solutions. Perhaps the most important problem of theoretical computer science is to determine whether these two classes are different: **P** and **NP**, where most of the scientist believe there are problems in NP not that are not included in **P**. There is an important subclass of **NP** problems, called NP-complete problems (**NPC**). Any **NPC** problem is at least as hard as all other problems in **NP**. It means that an algorithm to solve a specific **NPC** problem can be adapted to solve any other problem in **NP**. If **P=NP**, then it follows that no NP-complete problem can be efficiently solved on classical computer. It is not known whether quantum computers can be used to quickly solve all the problems in **NP** (although they can be used to solve some of them, e.g., factoring, which is believed not to be in **P**). Another important class in **PSPACE**. It consist of problems which can be solved using resources which are few in spatial size, but not necessary in time. **PSPACE** is believed to be strictly larger than P and NP (see Fig. 2.6), although this has never been proved. Finally, we mention **BPP** complexity class containing problems that can be solved using randomized algorithms in polynomial time, if a bounded probability of error is allowed.

In quantum complexity classes, we can define **BQP** (quantum, polynomial time) to be the class of all computational problems which can be solved efficiently on a quantum computer, allowing a bounded probability of error. Quantum computers

Table 2.1: an Example for 2-SAT problem

Assignment	Number of Conflicts c	Corresponding Integer
000	1	0
001	0	1
010	1	2
011	1	3
100	1	4
100	0	5
100	0	6
111	0	7

run only probabilistic algorithms [17], so **BQP** on quantum computer is the counterpart of **BPP** on classical computers. Exactly where **BQP** fits with respect to **P**, **NP** and **PSPACE** is not known. What is known is that quantum computers can solve all the problems in **P** efficiently, but there are no problems outside of **PSPACE** which can be solved efficiently, therefore **BQP** probably lies somewhere between them. An important implication is that if it is proved that quantum computers are strictly more powerful than classical computers, then it will follow that P is not equal to **PSPACE**. Although quantum computers may be faster than classical computers they can't solve any problems the classical computers can't (given enough time and memory). Since a probabilistic Turing machine can simulate quantum computers, they could never solve an undecidable problem like the halting problem [17].

2.4 Satisfiability problem (SAT)

NP search problems have exponentially many possible states and a procedure that quickly checks whether a given state is a solution [81]. Constraint satisfaction problems (CSPs) [88] are an example. A CSP consists of n variables, $V_1, V_2, \ldots, V_n$, and the requirement to assign a value to each variable to satisfy given constraints, where an *assignment* specifies a value for each variable.

One important CSP is the satisfiability problem (SAT), which consists of a logical propositional expression with n variables and the requirement to find a value (true or false) for each variable that makes the formula true. This problem has 2^n possible

assignments.

The k-satisfiability problem (k-SAT) is a combinatorial search problem, whose instance is a Boolean expression written using AND, OR, NOT, n variables, and m clauses. A clause is a logical OR of k variables, each of which may be negated. Given an expression, the solution is an assignment, i.e., a value of TRUE or FALSE values for each variable that will make the entire expression true, i.e., satisfying all the clauses [80]. For $k \geq 3$ these problems are NP-complete complete, i.e., among the most difficult NP problems in the worst case [81].

An example for a clause of k=3, with the third variable negated, is V_1 OR V_2 OR (NOT V_3), which is False for exactly one assignment for these variables:$V_1 = false$, $V_2 = false$, and $V_3 = true$. A clause with k variables is false for exactly one assignment to those variables, and true for the other choices of $2^k - 1$ possible assignments. Since the formula is a conjunction (AND) of clauses, a solution must satisfy every clause. We say an assignment conflicts with a particular clause when the values the assignment gives to the variables in the clause make the clause False.

For example, in a four variable problem, the assignment $V_1 = false$, $V_2 = false$, $V_3 = true$, and $V_4 = true$ conflicts with the $k = 3$ clause given above, while $V_1 = false$, $V_2 = false$, $V_3 = false$, and $V_4 = true$ does not. Thus each clause is a constraint that adds one conflict to all assignments that conflict with it. Then number of distinct clauses m is the number of constraints in the problem.

The assignments for SAT can also be viewed as bit strings with the correspondence that the i^{th} bit is 0 or 1 according to whether v_i is assigned the value $false$ or $true$, respectively. Also this bit strings could be the binary representation of integers, ranging from 0 to $2^n - 1$. For example, the assignment $V_1 = false$, $V_2 = false$, $V_3 = true$, and $V_4 = false$ represents the integer whose binary representation is 0010,i.e, the number 4.

An example for 2-SAT problem with n=3 is the propositional formula

$$(V_1 OR (NOT V_2)) AND (V_2 Or V_3) \tag{2.22}$$

This problem has a 8 possible assignments as shown in Table. 2.1. This problem have 4 solutions, i.e., The assignments with conflict number $c = 0$, Specifically, the

assignments with the bit representations 001, 101, 110 and 111. The remaining assignments with $c > 0$ are non-solutions, where they don't satisfies all the clauses.

Chapter 3

Adiabatic Quantum computation

3.1 Introduction

The model of adiabatic quantum computation is based on a theorem known as the quantum adiabatic theorem [9]. Informally, this theorem says that if we take a quantum system whose Hamiltonian slowly changes from initial Hamiltonian H_1 to final Hamiltonian H_2, then, under certain conditions, the ground (lowest energy) state of H_1 gets transformed to the ground state of H_2. This theorem is used to construct a new series adiabatic algorithms for solving optimization problems, in the following way. We take a Hamiltonian $H1$ whose ground state $|\psi\rangle$ is easy to construct, and a Hamiltonian H_2 whose ground state corresponds to the solution of the problem to be solved. Then, starting a quantum system in the state $|\psi(0)\rangle$ and slowly changing the Hamiltonian from H_1 to H_2 will solve our optimization problem. The adiabatic quantum algorithm have been applied to solve various optimization problems, such as satisfiability problems [25] [26], and set partitioning problem [28]. Later, the adiabatic version of various quantum algorithm were presented such as adiabatic Grover search algorithm [29], and adiabatic Deutsch-Joza algorithm [31]. where it was found that in some cases the adiabatic algorithm is not efficient, such as the adiabatic Grover algorithm and adiabatic Deutsch-Jozsa algorithm as they results in a complexity of $\mathrm{O}(N)$, which is equal to the classical counterpart. Later, the concept of local adiabatic evolution was presented [82], where it was proved to improve these algorithms performance giving an optimal performance of $\mathrm{O}(\sqrt{N})$ [82]. Also it was established that the adiabatic model is polynomially equivalent to

the slandered model of quantum circuits [36].

3.2 The Adiabatic Theorem

The adiabatic theorem is an important theorem in the thermodynamics. Informally, It states that a physical system that is initially in its ground state, tends to stay in this lowest energy state (ground state), provided that the Hamiltonian of the system is changed slowly enough, and if there is non zero gap between the ground state eigenvalue and the reset of the higher eigenvalues (higher energy states).

The term adiabatic means in thermodynamics to identify any isolated process, i.e., processes without the exchange of heat between the system and environment [17].

At a given time t, let $|\psi(t)\rangle$ denote the state vector of the system under the influence of the Hamiltonian $H(t)$. This quantum system evolves according to the Schrődinger equation [84] (we use $\hbar = 1$ for simplicity)

$$i\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle. \tag{3.1}$$

where the Hamiltonian of a quantum system gives a complete specification of the time evolution of this system.

The Schrődinger equation can also be described with reference to the unitary transformation U that is defined by the Hamiltonian $H(t)$ as

$$\frac{d}{dt}U(t)|\psi(0)\rangle = -iH(t)U(t)|\psi(0)\rangle, \tag{3.2}$$

thus with with the initial condition $U(0) = I$. we can say that The Hamiltonian evolution from $H(0)$ to $H(T)$ induces the unitary transformation $U(T)$. which can easily be expressed as $U = e^{-iTH}$ [84](by integrating the Eq. (3.2) over limits [0,T]).

Assume we can build a Hamiltonian $H^{(c)}$ with ground state encodes the solution of the problem instance to be solved, and prepare the system in the known ground state of another Hamiltonian $H^{(0)}$ (Usually we use a uniform superposition). Then the continuous adiabatic evolution of the system can be described by following the

time dependent Hamiltonian

$$H(t) = (1 - \frac{t}{T})H^{(0)} + \frac{t}{T}H^{(c)}. \tag{3.3}$$

with t ranging from 0 to T. Under suitable conditions, i.e., with a nonzero gap between relevant eigenvalues of $H(f)$, and with sufficiently enough time T, the adiabatic theorem grantees that the evolution maps the ground state of $H^{(0)}$ into the ground state of $H^{(c)}$. Then a subsequent measurement after time T gives the solution encoded at the ground state of $H^{(c)}$. The successful adiabatic evolution of the system is found to critically depends on the time of the evolution T [17].

3.3 The Quantum Adiabatic Algorithm

Consider a Quantum system evolves according to the Schrődinger equation (we use $\hbar = 1$ for simplicity)

$$i\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle. \tag{3.4}$$

where $|\psi(t)\rangle$ denote the state vector of the system under the influence of the time-dependent Hamiltonian $H(t)$. The Quantum adiabatic algorithm relies on the adiabatic theorem to to the computation, where it the Hamiltonian $H(t)$ varies slowly enough for time T, the state of the system will stay close to the instantaneous ground state of the Hamiltonian at each time t [68]. Assume we can build a Hamiltonian $H^{(c)}$ with ground state encodes the solution of the problem instance to be solved, and prepare the system in the known ground state of another Hamiltonian $H^{(0)}$. Then the adiabatic algorithm [68] can continuously evolve the state of the quantum computer using the time dependent Hamiltonian

$$H(t) = (1 - \frac{t}{T})H^{(0)} + \frac{t}{T}H^{(c)}. \tag{3.5}$$

with t ranging from 0 to T. Under suitable conditions, i.e., with a nonzero gap between relevant eigenvalues of $H(f)$, and with sufficiently enough time T, the adiabatic theorem grantees that the evolution maps initial state vector $|psi(0)\rangle$ which is the ground state of $H^{(0)}$ to the final state vector $|\psi(T)\rangle$ which will be close to the

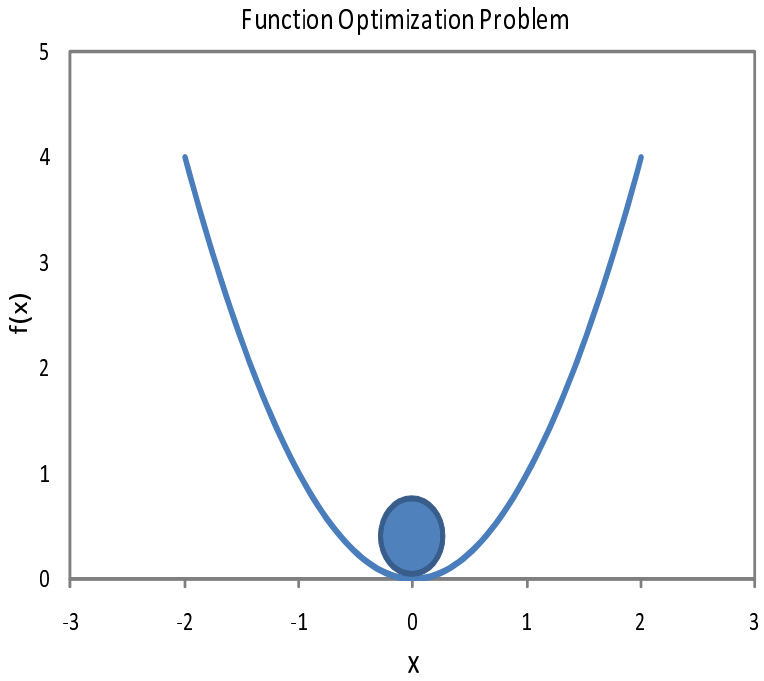


Figure 3.1: The evolution function $f_s(x)$ at s=0, i.e., equal to the initial function $f_0(x)$ [53].

ground state of $H^{(c)}$, thus a subsequent measurement after the evolution for time T gives the solution encoded at the ground state of $H^{(c)}$.

3.3.1 Example for function Optimization

Consider an optimization problem in which we need to find the x that minimizes the following function

$$f(x) = x^4 - 2x + 1. \tag{3.6}$$

for the interval of $x \in$[-2,2].

To be able to use the concept of the adiabatic method we do the following

1. Chose easy initial form of the function (easy to solve), here we chose it to be $f_0(x) = x^2$. giving the function in fig . 3.1.

2. Construct the evolving function (Hamiltonian) from the initial function $f_0(x)$

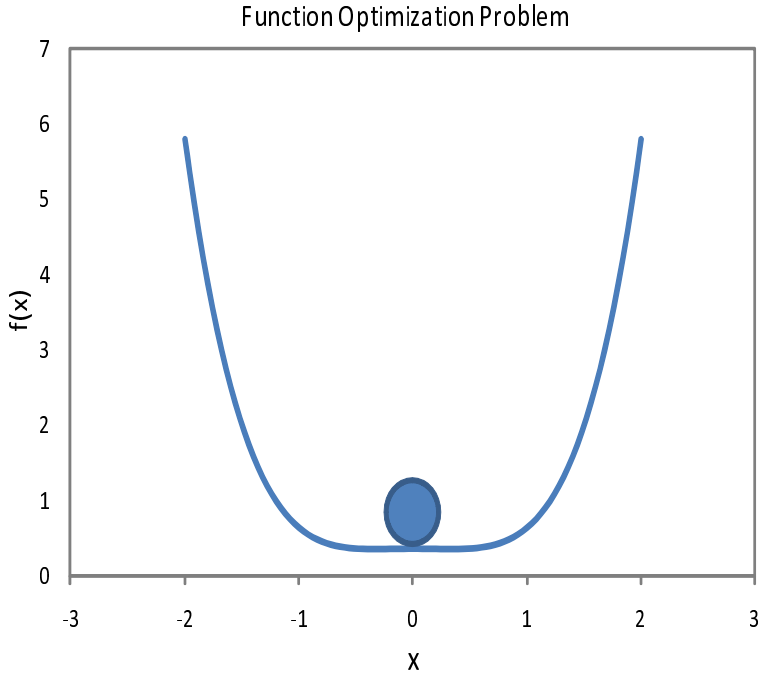


Figure 3.2: The evolution function $f_s(x)$ at s=0.32 [53].

and target function (the problem to be solved) $f_1(x)$ as

$$\begin{aligned} f_s(x) &= (1-s)\cdot f_0(x) + s\cdot f_1(x) \\ &= sx^4 + (1-3)x^2 + s. \end{aligned} \tag{3.7}$$

where s is the step parameter for the evolution with values changes slowly from 0 to 1.

3. Evolve the function $f_s(x)$ gradually from $f_0(x)$ to $f_1(x)$ with slow changing in s from 0 to 1. At s=1 system will be at the minimum of the function $f_1(x)$, i.e., we can the desired solution as shown in the figures 3.2,3.3, and 3.4.

3.4 The Minimum Gap

Minimum Gap G plays an important role in the quantum adiabatic evolution, where adiabatic theorem states that the evolution from the initial ground state to final ground state be successful provided that the evolving Hamiltonian energies (eigenvalues) never cross, i.e., with non-zero gap between the relevant eigenvalues of the system Hamiltonian H.

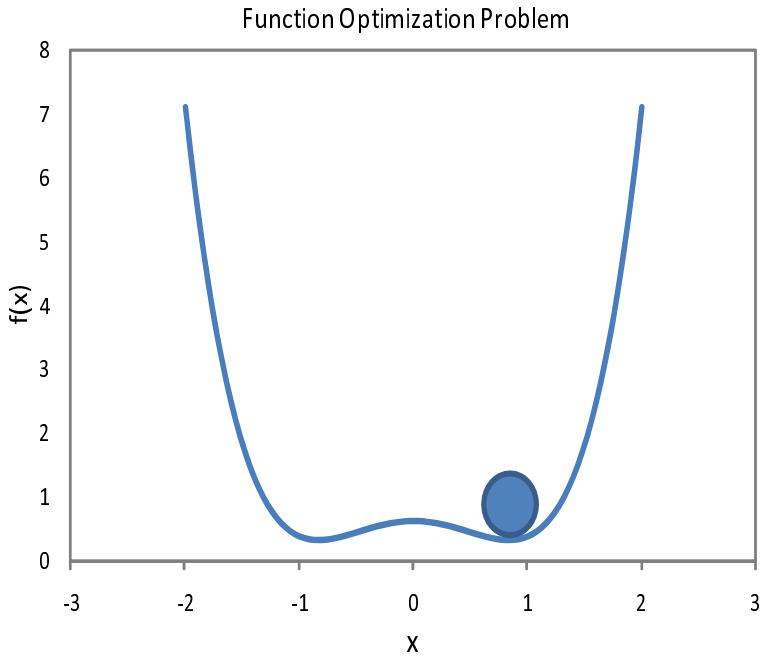


Figure 3.3: The evolution function $f_s(x)$ at s=0.62 [53].

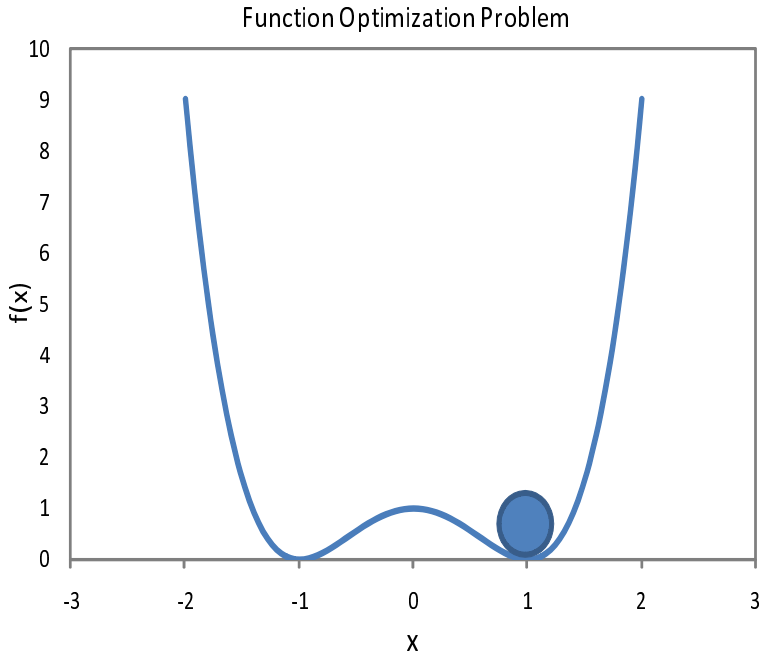


Figure 3.4: The evolution function $f_s(x)$ at s=1, i.e., at the targeted function $f_1(x)$, where the minimum solution is presented after all the evolution take place [53].

The system Hamiltonian which govern the Adiabatic quantum evolution could be generally presented as

$$H(t) = (1 - s(t))H^{(0)} + s(t)H^{(final)}. \tag{3.8}$$

with $s(t)$ changing from 0 to 1.

The adiabatic algorithm is considered to be successful if the required running time grows polynomially in the number of bits n. This time can be thought of the algorithm complexity (cost). The required running time is dominated by spectrum of the evolving Hamiltonian $H(t)$ [38], in particular the difference between the two lowest eigenvalues $E_0(t)$ (ground state) and $E_1(t)$ (the first higher energy level). where required running time T must obey

$$T = \frac{\varepsilon}{G^2}. \tag{3.9}$$

where G is the minimum gap given by

$$G = min(E_1(t) - E_0(t)) \tag{3.10}$$

and ε is less than the largest eigenvalue of $H^{(final)} - H^{(0)}$, and always polynomial. From all above we can conclude that if the evolving Hamiltonian $H(s)$ has an exponentially small minimum gap with respect to the number of qubits used in computation, then the corresponding algorithm is expected to be inefficient, whereas minimum gap which scales inverse polynomially gives an efficient quantum adiabatic algorithm whose running time is also polynomial [38].

3.5 Global vs Local Adiabatic Evolutions

When proposed the paradigm of quantum adiabatic evolution to design new series of quantum algorithms known as adiabatic quantum algorithms. The adiabatic quantum algorithms proposed by Farhi et al. [24] have been applied to solve various optimization problems, such as satisfiability problems [25] [26], and finding cliques in random graphs [27], where it has shown to give a polynomial search cost growth on average. In these algorithms, the condition for adiabaticity is fulfilled globally

by using only the minimum energy gap between ground and first exited state to determine the computation time. This method (global evolution) is found to be not efficient in some cases, such as adiabatic Grover's search algorithm [29] and adiabatic Deutsch-Jozsa algorithm [31] as they result in a complexity of $O(N)$ (which is complexity of classical algorithms). Therefore the need for an improvement rosed the idea of local adiabatic evolution [82], which proved to give an optimal performance of $O(\sqrt{N})$.

In the following sections we will show a brief definition for the Global adiabatic evolution and the local adiabatic evolution

3.5.1 Global Adiabatic Evolution

In the global adiabatic evolution scheme, s is changed uniformly with time ($s =$ constant) and the computation time scales as $1/g^2$. After evolution under $H(s)$ for a time T, the system is found in the ground state of The final Hamiltonian H^{final} with probability $(1-\varepsilon^2)^2$, provided the evolution rate satisfies [17]

$$\frac{|\langle \frac{dH}{dt} \rangle_{1,0}|}{G^2} \leq \ \varepsilon, \langle \frac{dH}{dt} \rangle_{1,0} = \langle E_1; t | \frac{dH}{dt} | E_0; t \rangle | \tag{3.11}$$

where $\varepsilon << 1$. The above formula (Eq. (3.11)) follows directly from adiabatic theorem by applying first order perturbation theory on a two-level system of relevant states. In particular, Eq. (1) implies that the minimum gap cannot be smaller than a certain value if we require the state at time t to differ from instantaneous ground state by a negligible amount (a smaller gap implies a higher transition probability to the first excited state). As long as the gap is finite, for any finite and positive, the time of evolution can be finite [17].

3.5.2 Local Adiabatic Evolution

The scheme of local adiabatic evolution was presented in order to improve the adiabatic evolution complexity (running time) which have shown the same of classical counterpart in some cases [29]. The idea is to impose a limit on the evolution rate during the whole computation by dividing the entire time interval T into infinitesimal time intervals dt and applying adiabaticity condition locally to each of these

intervals [82] [17], thus we can vary the evolution rate continuously in time and thereby speeding up the computation. Then a new local evolution condition would be

$$|\langle \frac{ds}{dt} \rangle \leq \varepsilon \frac{g^2(t)}{|\langle \frac{dH}{ds} \rangle_0, 1|} \tag{3.12}$$

Although, the scheme of local evolution improved the evolution where it could change the scaling time from from g_m^{-2} to g_m^{-1} (g_m is the minimum gap). However, it's response to the deliverance (sensitive to environmental noise) is a known disadvantage. In contrast, the global evolution is robust against the decoherance. Beside that, the local adiabatic evolution requires the knowledge of the spectrum which is not feasible for general Hamiltonian [17].

3.6 Practical Implementation of Adiabatic Quantum Computer

For the practical (actual) implementation [17], the Hamiltonian of the evolving system is discretized [82] in order to recast the adiabatic evolution in terms of unitary operators, Then the unitary operator U can be discretized to be as follow

$$U = \prod_{m=0}^{M} U_m \tag{3.13}$$

where m is current discrete step and M is the total number of discrete steps, U_m is the unitary transform for the step m and can be presented as

$$U_m = exp(-i[1 - \frac{m}{M} H^{(0)} + \frac{m}{M} H^{(final)}]\Delta t) \tag{3.14}$$

where

$$\Delta t = T/(M+1) \tag{3.15}$$

Discretization of a continuous Hamiltonian is straightforward process and changes the total run time T of the adiabatic evolution only polynomially. The required adiabatic limit is achieved when sufficient number of discrete steps M achieved.

The first experimental implementation of Actual adiabatic quantum evolution was for Shor's algorithm, it was demonstrated by Vandersypen et. al. [50] in 2001

using nuclear spins to find the prime factors of number 15. More recent experiments by Lu et. al. [51] and Lanyon et. al. [52] used photons as qubits and found the same factors. In 2005, Mitra et. al demonstrated the experimental implementations of local adiabatic evolution algorithms (Grover!s search and Deutsch-Jozsa algorithm) on a 2-qubit quantum information processor using NMR [54] [55]. Chuang et. al. [70] have demonstrated the implementation of a quantum adiabatic algorithm by solving MAXCUT 28 problem on a 3-qubit system by NMR. and most recently D-Wave systems company has announced the first quantum computer with 128 qubits capable to run adiabatic quantum algorithm to solve combinatorial search problems [39].

Chapter 4

Study on Quantum Heuristic Search in Knapsack problems

4.1 Introduction

In this chapter, we present an experimental study on the optimization behavior of the quantum heuristic search algorithm (QHS) for knapsack problem. QHS belongs to the gate model of the quantum algorithms, it could operate on superposition of all possible search states, and attempts to find the state with low cost. The cost c associated with each state is used to adjust the phase of the states amplitude, and a problem independent mixing operation combines the amplitude from different states. The experiments are carried out using random instances for different types of the knapsack problems for different number of bits, we uses two kinds of constrain handling methods for knapsack problems known as penalty function and random repair. The experimental results are compared with the conventional heuristic, the Genetic Algorithm (GA) using same instances.

4.2 The optimization algorithms

4.2.1 Quantum Heuristics search (QHS)

Quantum Heuristic Search [59, 60] is an alternative quantum algorithm for combinatorial search. The algorithm operates with superpositions of all possible search states for the problem instance. Each of their steps consists of adjusting the phases of the amplitudes in the super positions based on the properties of the problem

being solved, combined with problem-independent operation to mix the amplitudes among different states. A single trail of the algorithm can be presented as follows,

1. Initialize the amplitude equally for all 2^n states ,giving the initial state vector

$$|\psi_s^{(0)}\rangle = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{4.1}$$

   where $N = 2^n$.

2. For steps 1 through j, effect on the state vector using operator P to shift the amplitude phase based on the costs c associated with the states, and then mix them musing U operator. These operations is matrix multiplication as

$$|\psi_s^{(j)}\rangle = U^{(j)}P^{(j)}\cdots U^{(1)}P^{(1)}|\psi_s^{(0)}\rangle \tag{4.2}$$

3. Measuring the final superposition, giving the state s with probability $P(s) = |\psi^{(j)}|^2$. Thus the probability to obtain the minimum-cost state with a single trial is $P = \sum_s p(s)$, where the sum is over those s with minimum cost.

For each step h the phase shift matrix $P^{(h)}$, is a unitary diagonal matrix depending on the problem instance we are solving, with values determined by the cost associated with each state s as

$$P_{ss}^{(h)} = e^{i\pi\rho_h c(s)} \tag{4.3}$$

where ρ_h is a constant and $c(s)$ is the cost associated with search state s.
The mixing matrix U is problem independent given by $U^{(h)} = WT^{(h)}W$, where for state r and s Walsh transform W is given by $W_{rs} = 2^{-n/2}(-1)^{|r\wedge s|}$, where $|r \wedge s|$ is the number of 1 bits the both states have in common. The matrix $T^{(h)}$ is diagonal with elements depends on b, where b is the number of 1-bits state s contains as follow

$$T_{ss}^{(h)} = e^{i\pi\tau_h b} \tag{4.4}$$

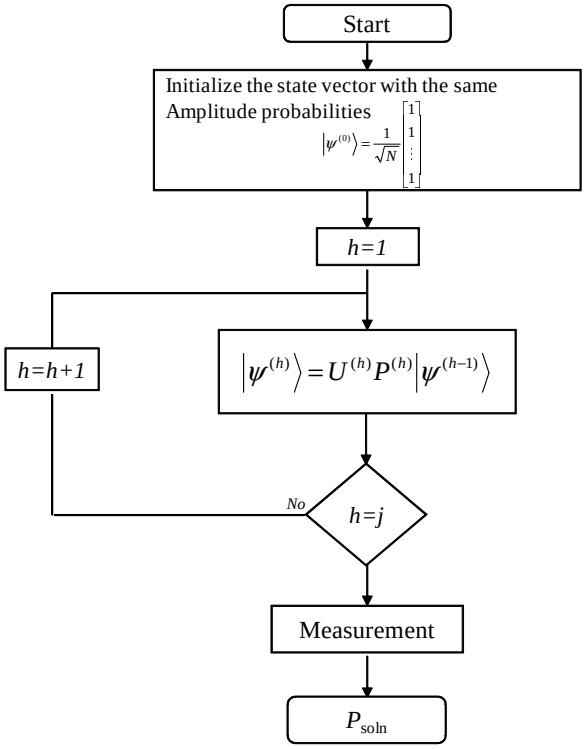


Figure 4.1: Flowchart represent single trial of QHS

where τ_h is a constant depending on the class of problem and the number of steps j, the mixing matrix U depend only on the hamming distance between the states $d(r, s)$,i.e, the number of bits with different values in the two states.That is, we can write $U^{(h)} = (-\tan(\pi\tau_h/2))^{d(r,s)}$, up to an over all phase and normalization constant [59] . The key of designing any quantum algorithm is the need to increase the probability of finding certain state (solution state, matching state) before the measurement operation, so that it could be the output after measurement.

4.2.2 Genetic Algorithm(GA)

Genetic algorithms are a part of evolutionary computing, which inspired by Darwin's theory about evolution. Simply we can say, solution to a problem solved by genetic algorithms is evolved [61], GA uses operators inspired by evolutionary biology such as mutation, selection, and crossover. It uses three basic parameters crossover probability and mutation probability, beside the population size [62]. A single trial of the GA can be described as follows:

1. Initialization: create a (random) starting population and evaluate fitness of each individual.
2. Recombination: recombine individuals using crossover and mutation.
3. Evaluation: evaluate fitness.
4. Selection: select best ones for the next generation
5. Termination: if the termination condition is satisfied, stop, otherwise continue at step 2.

Choosing appropriate values for the parameters in speed up the algorithm and prevent falling into local extreme

4.3 Comparative study between QHS and GA

4.3.1 Preparation

50 problems instances were randomly generated for each of the three classes of 0-1 knapsack problem with different number of bits, these instances were used in the evaluation experiments in order to observe the performance of QHS compared to GA. Two different constraint handling methods are used and the knapsack capacity is set to be half of the sum total of all items weights.
The phase parameters ρ_h and τ_h of QHS is chosen to vary linearly [59, 60] with the number of steps j as follows,

$$\rho_h = (R_0 + R_1(1 - (h-1)/j))/j \tag{4.5}$$

$$\tau_h = (T_0 + T_1(1 - (h-1)/j))/j \tag{4.6}$$

where h is the current step ,j is the number of steps for which the algorithm will be run. Thus for instance the first step uses $\rho_h = (R_0 + R_1)/j$, which means as j increases, the ρ_h and τ_h values become small so the corresponding P and U matrices

become close to identity matrices,i.e., each step introduces only small changes in the amplitudes. The values of the parameters R_0, R_1, T_0, and T_1 are 4, -3.4, 0.5392, and 3.5105, respectively, these values are same values used in the previous work, where it gave the best performance [59] [60] [58] [63]. The cost c of each assignment s ("individual") is calculated as $cost(s) = maxfitness - fit(s)$. The parameters of GA are chosen as follows, mutation rate =0.005, the crossover rate = 70% , the roulette wheel selection as selection method and the termination condition is 100 generations.

4.3.2 Knapsack Problem

0-1 Knapsack problem is well known optimization problem belongs to the class of NP-hard problems [64][65], where we have a knapsack with finite capacity and set of n items, where each item i have a weight w_i and a profit p_i. The objective of the KP is to select a subset of items such that the total profit z is maximized, while the total weight b doesn't exceed the capacity of knapsack, it can be mathematically formulated as follows [66],

$$\begin{aligned} &Given: \ c, w_i, p_i \\ &Maximize: \ z = \sum_{i=1}^{n} x_i p_i \\ &Subject\ to: \ b = \sum_{i=1}^{n} x_i w_i \leq c \ and \ x_i \in 0, 1 \end{aligned} \tag{4.7}$$

Classes of KP

The classes of KP focusing on the relationship between Weight and Profit [67] as shown in Figures 4.2, 4.3, and 4.5 as follows,

Uncorrelated: The profit p_i and the weight w_i of the item i are uniformly random numbers in [1, v] .

Weakly correlated: The weight w_i is uniformly random in [1, v], and p_i is uniformly random in [max (w_i -r, 1), w_i +r]

Strongly correlated: The weight w_i is uniformly random in [1, v], and $p_i = w_i$ +r .

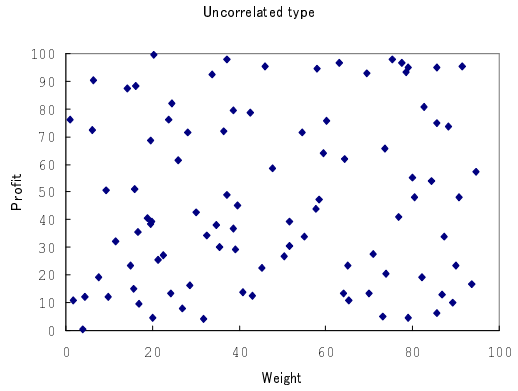


Figure 4.2: Uncorrelated (C_u).

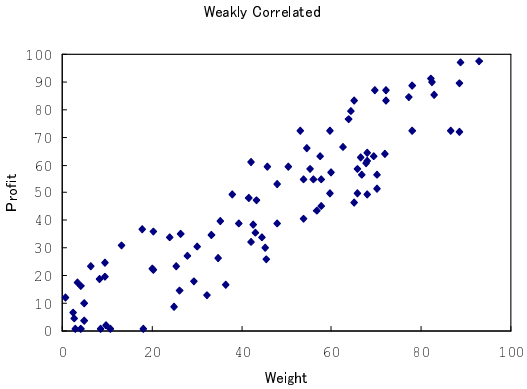


Figure 4.3: Weakly correlated (C_w).

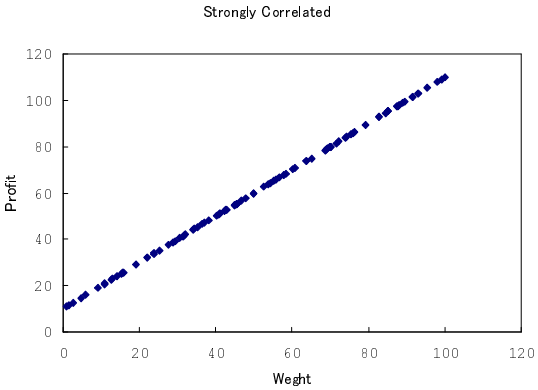


Figure 4.4: Strongly correlated (C_s).

4.3.3 Constraint Handling Methods

With the aim to evaluate the dependableness on the problem constraint handling methods in solving KP with QHS and GA, two different methods are used , Penalty function and Random repair [67] to handle the constraint violations,i.e., the excess of total weights of the chosen items the capacity of the knapsack .

1. **Penalty Function**

   Considering n items Knapsack problem with capacity given by $c = \frac{1}{2}\sum_{i=1}^{n} w_i$. The Fitness is calculated from a penalty function as follows:

   $$Fit(p_i) = \sum_{k=1}^{n} p_k x_{ik} - \alpha \cdot \max\{0, \sum_{k=1}^{n} w_i x_{ik} - c\}, \tag{4.8}$$

   Where p_i is the profit and $w_i$is the weight of the item i, C is the capacity of the knapsack and $\alpha = max_{k=1\ldots n}\{p_k/w_k\}$.

2. **Random Repair.**

   To evaluate the fitness using random repair method the following two steps will be applied:

   Step1: Repeat selecting an item in the knapsack randomly and removing it until the knapsack is filled.

   Step2: Repeat selecting an item and adding into the knapsack until capacity are exceeded. When the capacity is exceeded, the item added at the end is removed.

   The Step2 is applied even when weight limits are satisfied [67].

Penalty function method is a general-propose constraint handling method, however, eqn (4.8) depends on KP. Random Repair method is constraint handling method depending on KP. Therefore the dependableness on problem-specific constraint handling method is expected to be larger in Random repair method.

4.3.4 Experimental results

The experiments is carried out solving 10 different random instances each is averaged over 10 trials, QHS and GA are run for 100 generation(steps) for each trail. As result,

the rate of finding the optimal solution and search cost for the both algorithms will be compared.

Fig. 4.5 shows the rate of finding the optimal solution *vs.* number of steps of QHS and GA for the three classes of the KP-problem, Uncorrelated (C_u), weakly correlated (C_w), and strongly correlated (C_s) with different number of bits 8,16, and 20. It shows that the quantum algorithm QHS can find the optimal solution on average effectively with fewer number of steps when compared with GA for the 3 types, however with lower rate of finding the optimal solution. Also the fig indicate that the Strongly correlated type instances was the hardest to solve using QHS, where it gives lowest probability to find the solutions (from 0.44 to 0.52). That is because the QHS search performance is affected mainly by the distribution of the 1 bits inside the state (the assignment), which effects directly on the problem-independent operator U (which mix the amplitudes between the different states). Thus the results revealed that penalty function method couldn't improve the performance of QHS.

Fig. 4.6 shows the rate of finding the optimal solution *vs.* number of steps for QHS and GA for the same instances used in Fig. 4.5, however it uses random repair method as constraint handling method. Focusing on Figure 4.6 we can see that QHS could find the optimal solutions with rates higher than shown in Fig. 4.5 and also the required number of steps for QHS is below that of GA. The fig also shows the improvement of the rate of finding the optimal solution specially, in strongly correlated type with 20 items (62%) instead of (44%).

The comparison between experimental results shown in Fig. 4.5 and Fig. 4.6 have revealed that using the random repair method could improve the rate of finding the optimal solution of QHS than those of using penalty function method. This is mainly because random repair method redistribute the 1 bits inside the assignments (adding or removing items from knapsack), which improves the performance of the mixing operator U. As result, it leads to improve in the search performance of QHS.

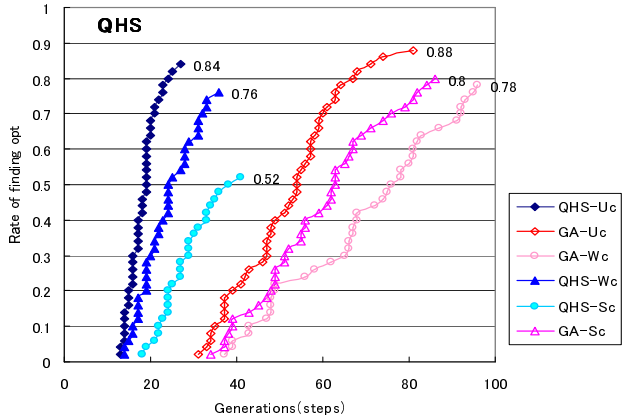
Fig. 4.7 compare the search cost of QHS *vs.* GA in C_u, C_w, and C_s with different number of bits 4,8,12,16, and 20. For each number of bit n the same number of instances is solved using the both algorithms, each point value is the average of 100 trials, and the search cost is calculated as total number of steps in the successive

trials over the number of successive trials .

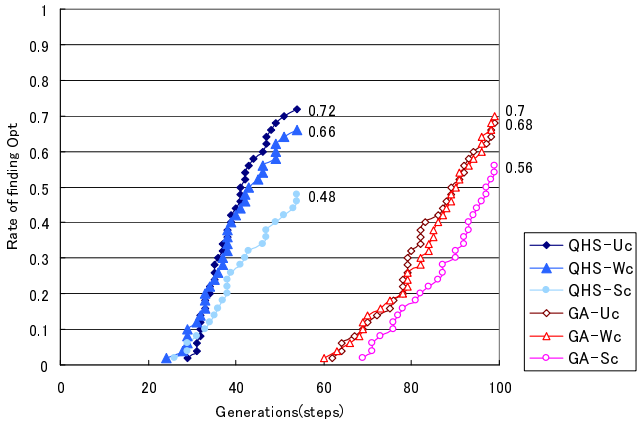
Fig. 4.8 also show the comparison of the search cost for QHS and GA in C_u, C_w and C_s using random repair method. Focusing in the Figures 4.7 and 4.8 we can see that QHS has search costs are below of the classical algorithm GA or at least comparable all the time, however, both algorithms search costs growth exponentially, the comparison between the average search cost of both algorithm measured by total number of steps in the successive trials over the search trials is the most significant comparison. This is because the actual search times will depend on detailed implementation of the steps. Although the number of elementary computational steps involving evaluating the cost is the same on the both algorithms, the difference in clock rates of classical and quantum machines remain to be seen.

4.4 Conclusion

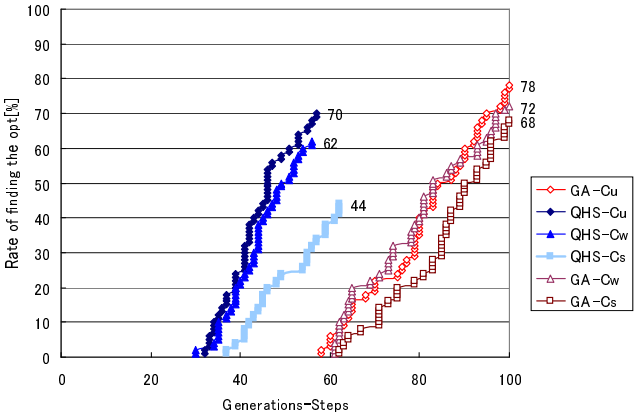
The experiments done with QHS with different constraint handling method for KP, and the comparison with GA indicate that the QHS which could operates with superpositions of all possible search states and use the cost structure of problem instances, on average has performance comparable to classical heuristics where number of steps required by QHS was lower than required by GA to find the optimal solution, also the experiments revealed that using the random repair method as constraint handling method improved the search performance of the QHS than using penalty function, specially, with the strongly correlated type. However, the search cost after using random repair method is increased than using the penalty function method but still comparable with GA. For future direction, including other properties used in classical heuristics may also give useful phase adjustment for example include how an assignment cost compare to those of its neighbors. Currently, such quantum machines do not exist. Instead, we must simulate the quantum algorithm on conventional machines, so each trial requires exponential cost and memory. Thus we are limited to investigating much smaller problems up to 20 variables, the simulation evaluates properties of all search states and so is considerably more expensive than evaluating conventional heuristics.



(a) QHS and GA in 8 items

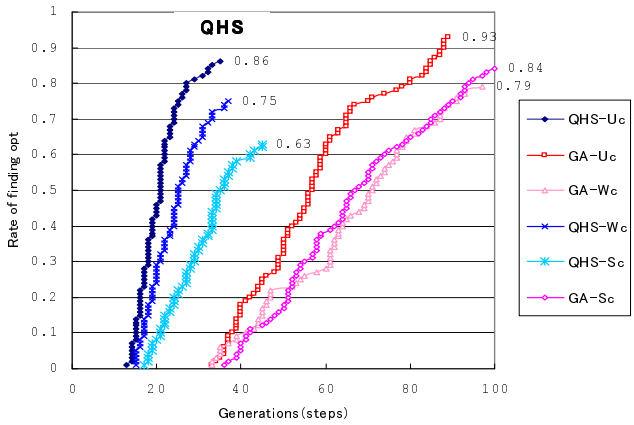


(b) QHS and GA in 16 item

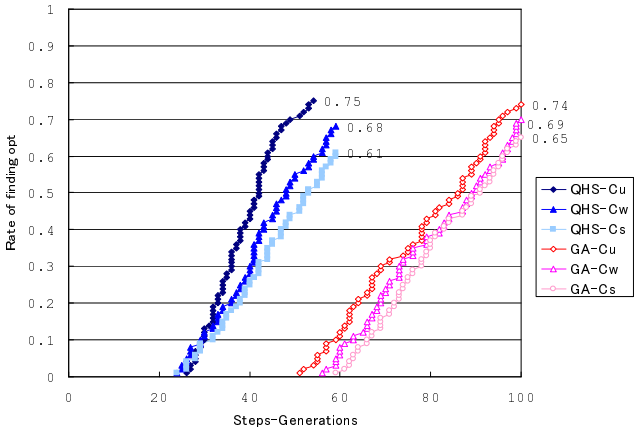


(c) QHS and GA in 20 items

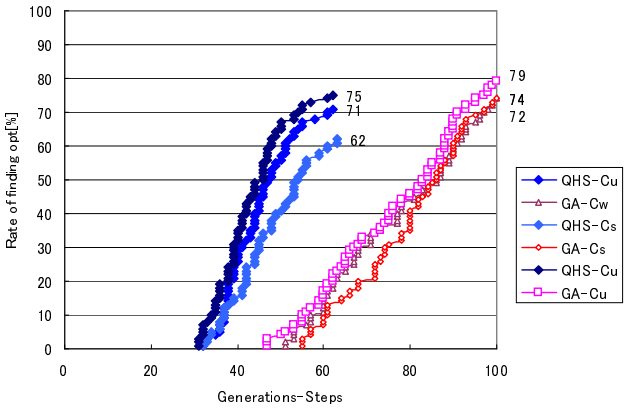
Figure 4.5: Comparison between QHS and GA using Penalty method on the rate of finding the optimal solution



(a) QHS and GA in 8 items



(b) QHS and GA in 16 item



(c) QHS and GA in 20 items

Figure 4.6: Comparison between QHS and GA using Repair method on the rate of finding the optimal solution

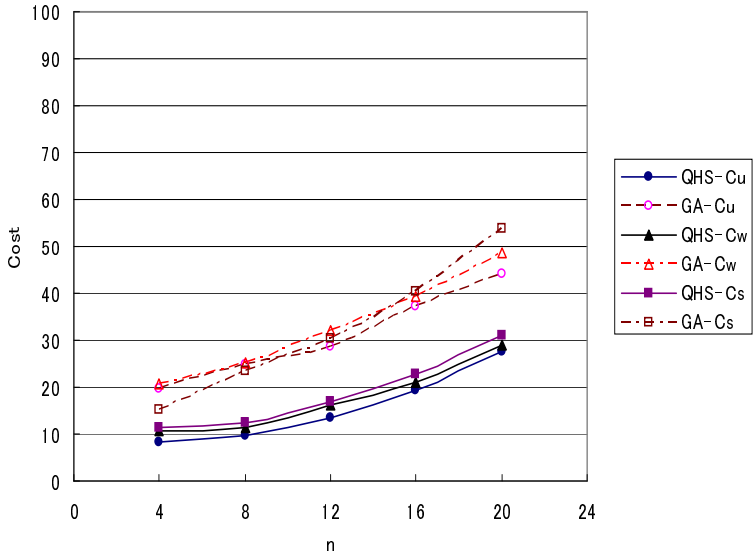


Figure 4.7: The comparison between the search cost for QHS and GA using Random repair

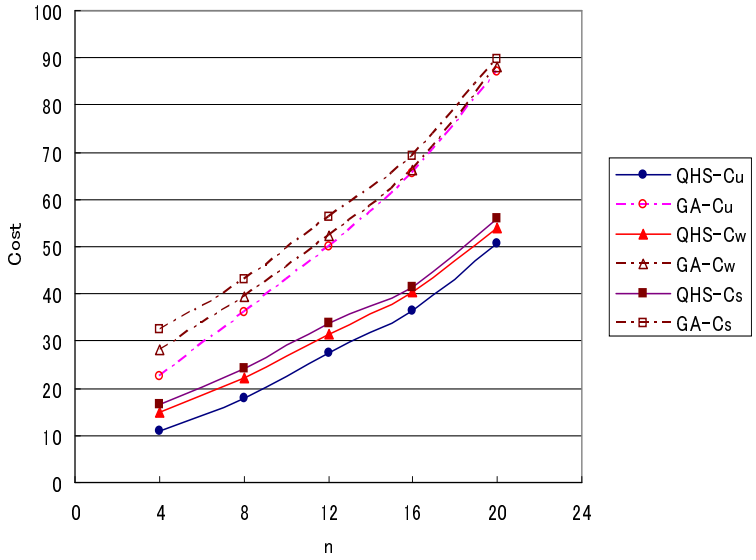


Figure 4.8: The comparison between the search cost for QHS and GA using Random repair

Chapter 5

Study on discrete adiabatic computation with quadric variation method

5.1 introduction

The model of adiabatic quantum computation is a new paradigm for designing quantum algorithms, proposed by Farhi et al. [24]. It relies on the adiabatic theorem to do the calculations, where the quantum computer evolves slowly for sufficient time T to switch gradually from an initial Hamiltonian with known ground state, to a final Hamiltonian whose ground state encodes the known solution [25]. Recently It was recently established that this model is polynomially equivalent to the standard model of quantum circuits [36] [84]. Nevertheless, this model provides a completely different way of constructing quantum algorithms and reasoning about them [79]. Therefore, it is seen as a promising approach for the discovery of a new quantum algorithms [24].

The adiabatic quantum algorithm have been applied to solve various optimization problems, such as satisfiability problems [25] [26], finding cliques in random graphs [27], and set partitioning problem [28], where it has shown to give a polynomial search cost growth on average.

This Chapter presents the first version of the quadric variation method, which is a partiality-monotonic variation method for the phase functions in quantum adiabatic algorithm. With the aim to show the improvement in the search behavior of the

quantum adiabatic algorithm using the proposed method when it is compared to the existing methods, the experimental study in solving random instances of 3-SAT problems with different number of bits has been carried out using the three variation methods, linear, cubic, and quadric, for which the corresponding search costs and probability of finding the solution are shown. The experimental considerations and results shown in this chapter are presented at The World Congress on Nature and Biologically Inspired Computing (NaBIC2010) and published in it's proceedings [69].

5.2 3-SAT Problem

The k-satisfiability problem (k-SAT) is a combinatorial search problem, whose instance is a Boolean expression written using AND, OR, NOT, n variables, and m clauses. A clause is a logical OR of k variables, each of which may be negated. Given an expression, the solution is an assignment, i.e., a value of TRUE or FALSE values for each variable that will make the entire expression true, i.e., satisfying all the clauses [80]. An example 2-SAT instance with 3 variables and 2 clauses is $(v_1 \mathrm{OR}(NOT v_2))\ AND\ (v_2\ \mathrm{OR}\ v_3)$, which has 4 solutions, for example, $v_1 = v_2 = false$ and $v_3 =$ true. For a given instance, the cost $c(s)$ of an assignment s is the number of clauses it does not satisfy.

For $k \geq 3$, k-SAT is NP-complete, i.e., among the most difficult NP problems in the worst case [80] [81]. With aim to show the average behavior of the algorithm, we generate random instances of 3-SAT problem, in which the m clauses are selected uniformly at random, .i.e., for each clause, a set of 3 variables is selected randomly. In this study we focus on the decision problem, i.e., finding a solution, where the adiabatic algorithm is probabilistic, so cannot definitively determine if the solution exists or not, thus we consider soluble instances only.

5.3 The discrete Adiabatic algorithm

Originally, the adiabatic method is continuous process. However, with aim to compare with other discrete methods, here we use the discrete formulation of the algorithm which proved to be algorithmically equivalent [84][68], it acts on the

amplitude state vector initially in the ground state of the Hamiltonian $H^{(0)}$, which can be represented for n qubits quantum system as $|\psi_s^{(0)}\rangle = \frac{1}{\sqrt{2^n}}[1, 1, ..., 1]^T$ [68]. Discretization of a continuous Hamiltonian is straightforward process and proved to change the total run time of the algorithm only polynomially [70].

Consider a discreet Hamiltonian $H(f)$ in the general form

$$H(f) = \tau(f)H^{(0)} + \rho(f)H^{(c)}, \tag{5.1}$$

where $\tau(f)$ and $\rho(f)$ are mixing and phase shift function, respectively. Both of them are functions of f $(0 \leq f \leq 1)$, shown in Table.1, subject to the boundary conditions

$$\tau(0) = 1, \rho(0) = 0, \tag{5.2}$$

$$\tau(1) = 0, \rho(1) = 1. \tag{5.3}$$

Although, the two functions $\tau(f)$, and $\rho(f)$ are not necessary to be monotonic [85] (i.e. obey the constraint $\tau(f) + \rho(f) = 1$), here we consider only the monotonic functions. In matrix form [68], the Hamiltonian $H^{(c)}$ could be presented as

$$H_{r,s}^{(c)} = c(s)\delta_{r,s}, \text{where} \quad \delta_{r,s} = \begin{cases} 1 & \text{if } r{=}s \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

This Hamiltonian introduces a phase shift factor in the amplitude of assignment s depending on its associated cost $c(s)$, where the higher cost results in more phase shift. The Hamiltonian $H^{(0)}$ can be implemented with elementary quantum gates by use of the Walsh-Hadamard transform with elements $W_{r.s} = 2^{-n/2}(-1)^{r \cdot s}$ [68], where $H^{(0)} = WDW$ and D is a diagonal matrix with the value for state r given by the sum of the bits, i.e, the element $D_{r,r}$ is just a count for the number of bits equal to 1 in state r.

A single trial of the algorithm, as shown in Fig. 5.1, consists of j steps with parameter Δ and can be described as follows [69]:

1. Initialize the amplitude state vector to the ground state of $H^{(0)}$ giving equal values for all states as $|\psi_s^{(0)}\rangle = \frac{1}{\sqrt{N}}[1, 1, ..., 1]^T$.

2. For steps $h = 1$ to j, repeat the matrix multiplication :

$$|\psi^{(h)}\rangle = U_h(f)|\psi^{(h-1)}\rangle. \tag{5.5}$$

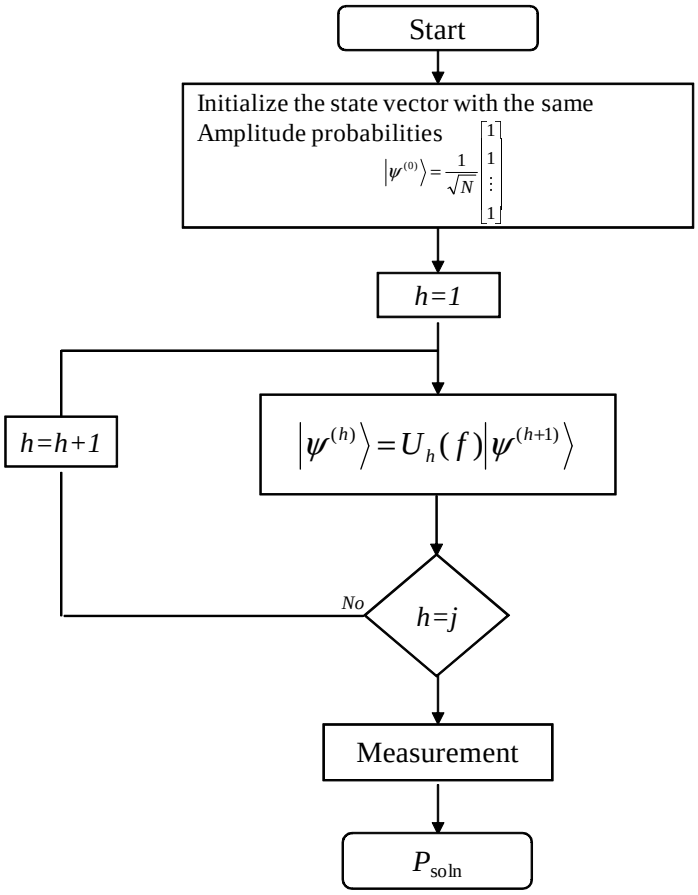


Figure 5.1: Flowchart represents the discrete version of adiabatic quantum algorithm

where $|\psi^{(h-1)}\rangle$ is the amplitude state vector at the step $h-1$, and $U_h(f)$ is unitary evolution operator for h^{th} step which can be represented as

$$U_h(f) = e^{-i\tau(f)H^{(0)}\Delta} \cdot e^{-i\rho(f)H^{(c)}\Delta}. \tag{5.6}$$

3. Measure the final system after the j steps take place. Then probability to find a solution is given by $P_{soln} = \sum_s ||\psi^{(j)}||^2$.

5.4 The Quadric Variation Method (Partial Monotonic version)

Recently several variation methods for phase shift $\rho(f)$, and mixing function $\tau(f)$ in the discrete adiabatic quantum algorithm are presented, such as linear variation [24], and cubic variation [68]. Here we propose a partially monotonic version of

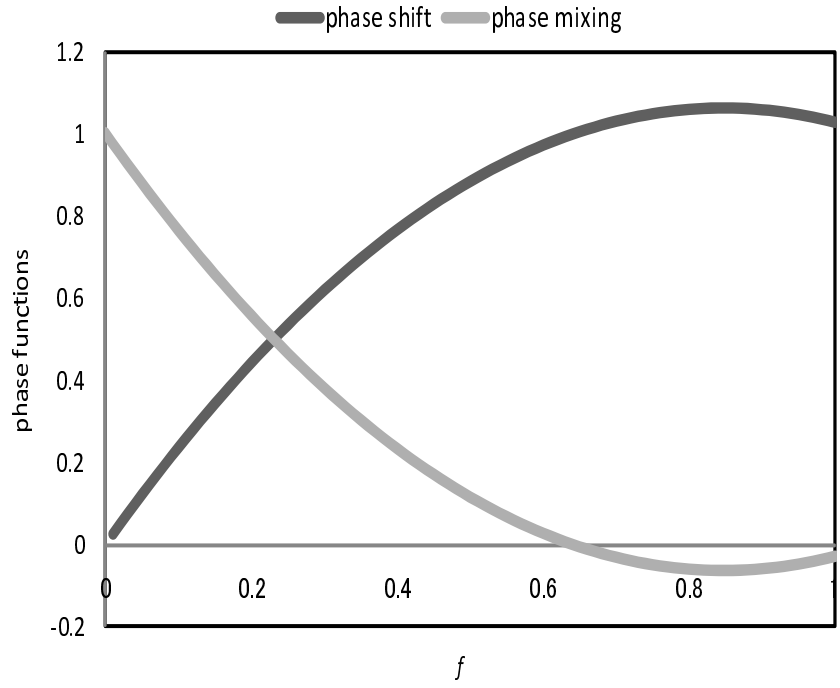


Figure 5.2: Phase shift and phase mix functions *vs.* f for the partial monotonic quadric variation method with $a \cdot f - b \cdot f^2$.

quadric variation method for $\rho(f)$ and $\tau(f)$ in attempt to decrease the over all resulting search cost and speed up the adiabatic algorithm. In this method $\rho(f)$ and $\tau(f)$ vary monotonically as quadric polynomial in f as $a \cdot f - b \cdot f^2$ and $1 - [a \cdot f - b \cdot f^2]$,respectively. This proposed formula is easy to construct, and it was found to satisfy the conditions (4) and (5) as well as the adiabaticity condition, i.e, using this method doesn't lead the relevant eigenvalues (energies) of the evolving Hamiltonian $H(f)$ to cross.

A word Monotonic means that it satisfies the condition ($\rho(f)+\tau(f)=1$) for all values of f, and have values bounded between 0 and 1. The values of a and b depend on the problem to be solved. However, these values are bounded by satisfying the conditions (4), and (5). For satisfiability problems these values are found to be a=2.5187 and b=-1.483. These values are found for a set of 100 random instances of 3-SAT problem with a best performance [69].

Fig. 5.2 shows the phase functions for quadric variation method *vs.* the f value

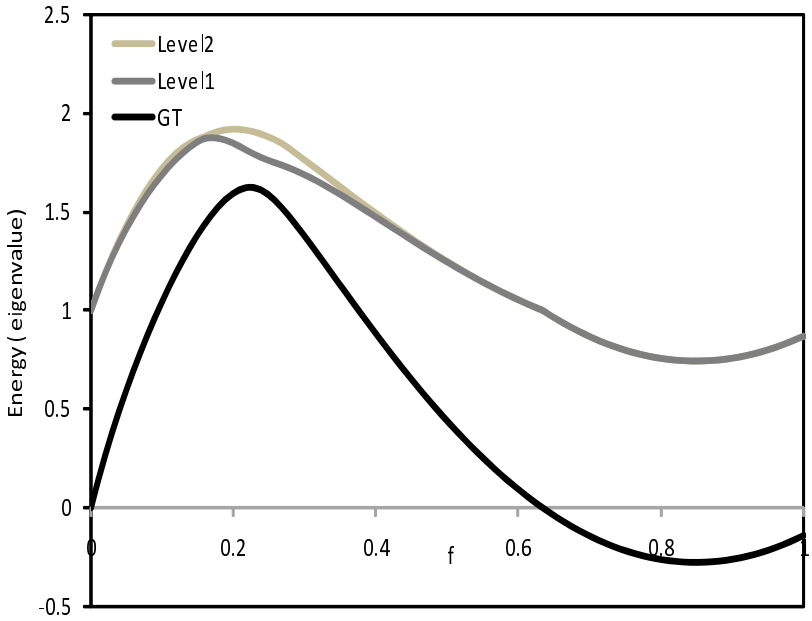


Figure 5.3: Lowest two 3 eigenvalues vs. f for an instance with n=8, and one solution. The black curve shows the ground state, gray curve and dark gray curve are level 1, level 2, energies, respectively.

from 0 to 1. From this figure we can see that the quadric variation method shown has small diversity area and big intensification area which gives a sign for expecting better results. Also we we can see that the values of $\rho(f)$ and $\tau(f)$ are not completely bounded between 0 and 1, thus we say this method is partially monotonic. However the values of a and b could be optimized depending on the problem to be solved to get the best probability of finding the solutions or to convert the quadric variation to be completely monotonic as we will show in the next chapter.

Fig. 5.3 shows the Lowest 3 eigenvalues (Energy levels) of the evolving Hamiltonian $H(f)$ $vs.$ f in solving an instance of 3-SAT problem with one solution. In this evolution the quadric variation method is used for the phase shift $\rho(f)$, and mixing τ functions. This figure shows that the eigenvalue of the ground state (bold black curve) and the next two smallest higher eigenvalues correspond to non solutions (gray and durk gray curve respectively), never cross, i.e., with non-zero gap. As a result this figure reveal that evolution under the discrete Hamiltonian $H(f)$ (see Eq.(4)) with the quadric variation method follows the adiabatic theorem.

5.5 The Recent Variation Methods

Recently several variation methods for phase shift $\rho(f)$, and mixing function $\tau(f)$ in the discrete adiabatic quantum algorithm such as linear variation [24], and cubic variation[68]. With aim to show the improvement of the proposed method we use linear [24], and cubic [68] methods described in next sections to compare the search behavior and performance of the adiabatic algorithm with proposed method.

5.5.1 Linear Variation Method

Linear variation method is corresponding method to the continuous version of the adiabatic algorithm (The original scheduling method for the adiabatic algorithm). It varies the phase shift $\rho(f)$ and mixing functions $\tau(f)$ of the adiabatic algorithm linearly as simple monotonic function as f and $1-f$, respectively. This method is easy to construct. However, it was proven that it needs large number of steps to reach high probability of finding solutions [69]. This method presents the original idea of the adiabatic evolution model to do a computations [24] [25]. This method is completely monotonic where it starfishes the condition ($\rho(f)+\tau(f)=1$) for all values of f and have values always bounded between 0 and 1.

Fig. 5.4 shows the phase functions for linear variation method *vs.* the f value from 0 to 1. From this figure we can note that the linear variation method has diversity area and intensification with equal size, which gives a sign for requiring an intensive phase shifting and mixing , i.e, requires higher number of steps to achieve high probability of finding a solutions.

Fig. 5.5 shows the Lowest 3 eigenvalues (Energy levels) of the evolving Hamiltonian $H(f)$ *vs.* f in solving an instance of 3-SAT problem with one solution using linear variation method. This figure shows that the eigenvalue of the ground state (bold black curve) and the next two smallest higher eigenvalues correspond to non solutions (gray and dark gray curve respectively), never cross, i.e., with non-zero gap. As a result this figure reveal that evolution under the discrete Hamiltonian $H(f)$ (see Eq.(4)) with the linear variation method obeys the adiabatic theorem.

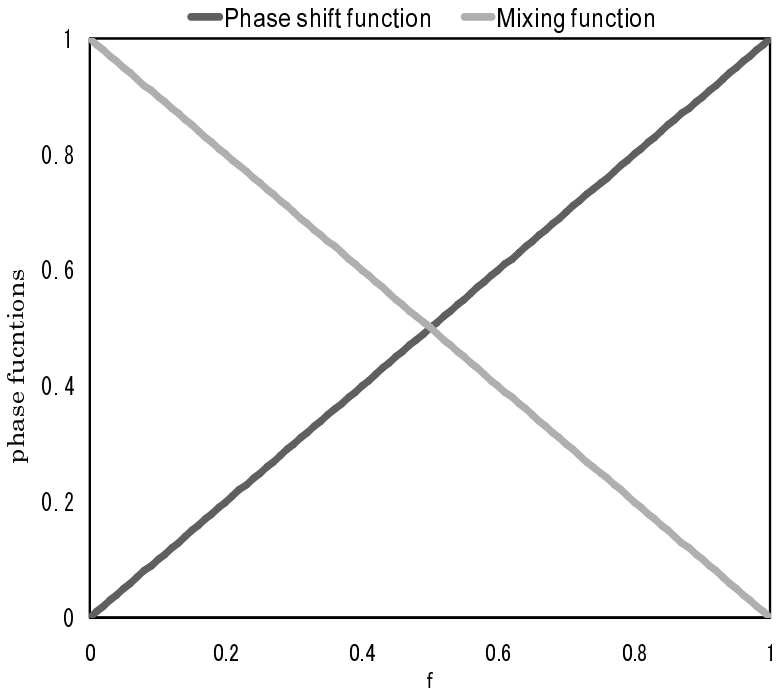


Figure 5.4: Phase shift and phase mix functions *vs.* f for the linear variation method.

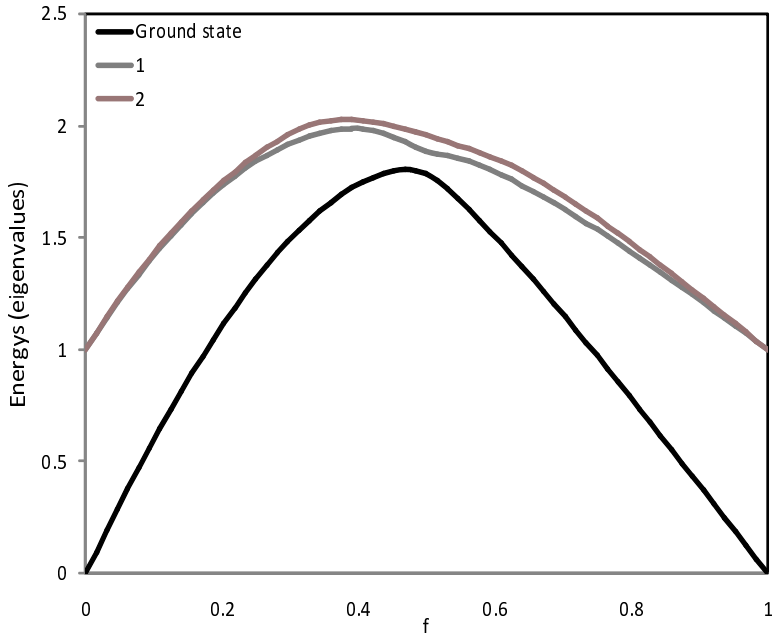


Figure 5.5: Lowest two 3 eigenvalues *vs.* f for an instance with n=8, and one solution. This instance was solved using the algorithm with linear variation method.

5.5.2 Cubic Variation methods

Cubic variation method is similar to the functional form optimizing the adiabatic method for unstructured search which proved to reduce the resulting cost [68] [82] [84]. In this method $\rho(f)$ and $\tau(f)$ vary monotonically as cubic polynomial in f as $1.921f - 2.665f^2 + 1.782f^3$ and $1 - [1.921f - 2.665f^2 + 1.782f^3]$, respectively. This method is completely monotonic where it satisfies the condition (ρ+τ=1) for all values of f and have values always bounded between 0 and 1. And also it satisfies adiabaticity condition, i.e, using this method never lead the relevant eigenvalues (energies) of the evolving Hamiltonian $H(f)$ to cross.

Fig. 5.6 shows the phase functions for cubic variation method *vs.* the f value from 0 to 1. From this figure we can note that the cubic variation method has diversity area and intensification with equal size(However smaller than linear variation method), which gives a sign for requiring an intensive phase shifting and mixing , i.e, it may require larger number of steps to achieve high probability of finding a solutions.

Fig. 5.7 shows the lowest 3 eigenvalues (Energy levels) of the evolving Hamiltonian $H(f)$ *vs.* f in solving an instance of 3-SAT problem with one solution using linear variation method. This figure shows that the eigenvalue of the ground state (bold black curve) and the next two smallest higher eigenvalues correspond to non solutions (gray and dark gray curve respectively), never cross, i.e., with non-zero gap. As a result this figure reveal that evolution under the discrete Hamiltonian $H(f)$ with the cubic variation method satisfies the adiabaticity condition.

5.6 The Simulator

5.6.1 Description

Simulating the quantum computer (Algorithm) is such a hard issue causes in expectational growth in required memory of the computer. This is due to represent a pure quantum principles such as superposition and parallelism, where any operation must be applied in once to all possible states in the superposition. In order to implement the parallelism the vector/ matrix representation are used for the simulator. For

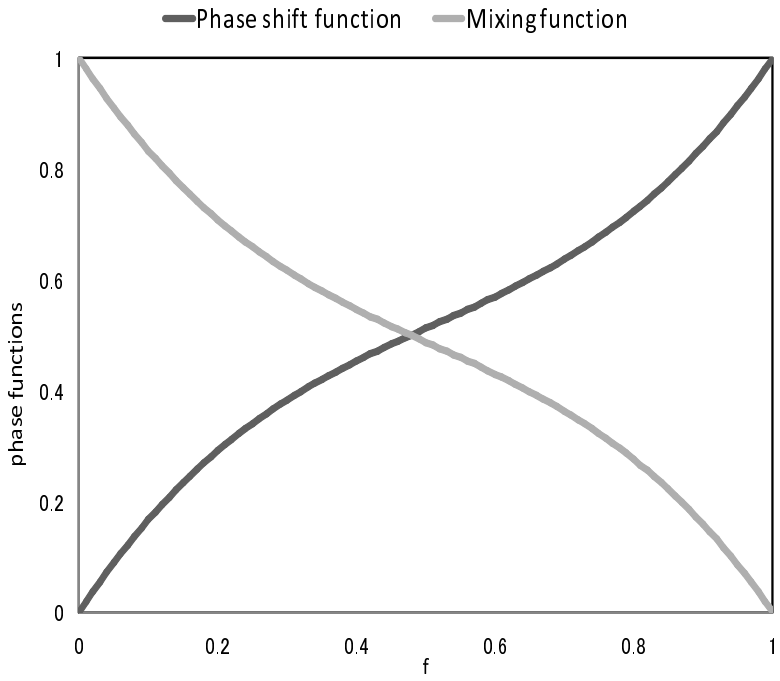


Figure 5.6: Phase shift and state mix functions *vs* f for the Cubic variation method.

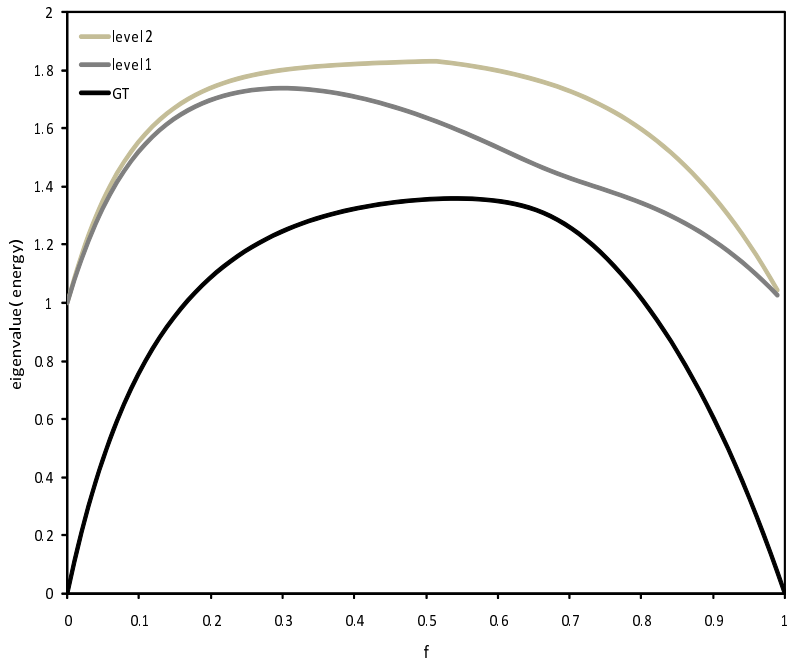


Figure 5.7: Lowest two 3 eigenvalues vs. f for an instance with n=8, and one solution. This instance was solved using the algorithm with cubic variation method.

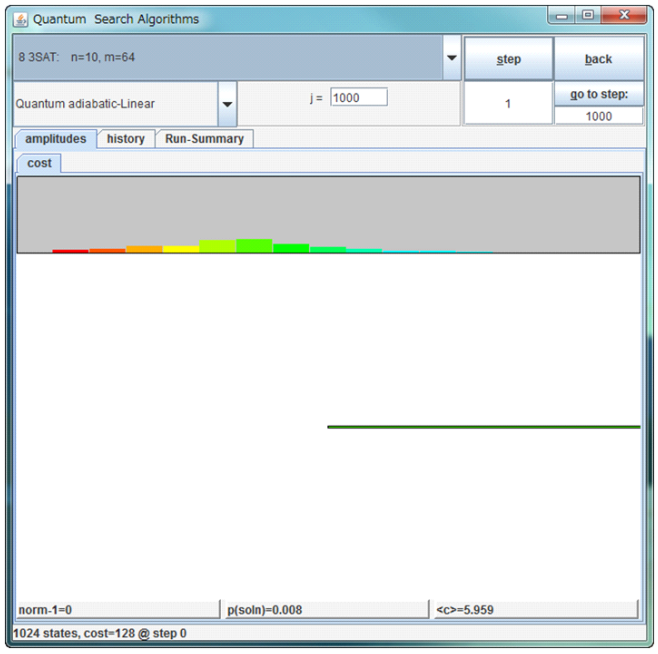


Figure 5.8: The simulator of Quantum search algorithms (The interface).

example, the satisfiability problem with $n = 3$ variables could be presented by state vector of length 2^n, i.e., with length 8 in this case where all possible states (8 states) could be presented, Also any operator effect on this state vector must be a matrix of $2^3 * 2^3$. The experiments presented in next sections are carried out using Core i7 processor with 8 cores and with 16 Giga of memory, which is currently the best available configuration.

Fig. 5.8 shows The simulator we use. It is created using a known computer programming language, Java, and effectively can simulate the search behavior of various Quantum Search Algorithms such as the quantum heuristic search (QHS), the Quantum Adiabatic Algorithm with different variation methods, and the Grover algorithm for database search.

5.6.2 Interface

The following Figures show the program interface

Fig. 5.9 shows the first tab of the interface, it shows the probability amplitude

of each state in the superposition in initialization state where all of them have the same real value, this amplitudes are shown as lines in a complex plane where X-axis is the real part and Y- axis is the imaginary part. Also in in this figure we can see that the how the program is easy to use where you can chose the problem instance to be solved as well as the method (the algorithm) from the combo box. Also in this Tab (amplitude tab) we can always now the Norm of the state vector, current probability of finding the solution, and the average conflict number from the labels marked as 1, 2, 3 , respectively, in the bottom part of the tab.

Fig. 5.10 shows the probability amplitude as lines in complex plane rotating as effect of phase shift operators and mixing operators, The black lines are the solutions.

Fig. 5.11 shows the second tab of the interface named the probability tab. It shows the probability of finding the solution as function of number of steps. Here you can compare various methods search behavior in solving the same problem instance as shown in the Fig. 5.11.

Fig. 5.12 shows the final tab named Run-summary. here you can get a summary of experimental results, it shows the algorithm name, resulting probability of finding the solution $P(solution)$, total number of steps, minimum search cost, and the step at maximum $P(solution)$.

5.7 The parameter Δ

A good performance of the discrete adiabatic algorithm requires an appropriate choice of parameter Δ. In the discrete formulation, Δ parameterizes the operators of Eq.(9) rather than determining the time T required to perform them [68]. By contrast, Δ in the continuous formulation determines the time to perform the operators as $T = j \cdot \Delta$. The recently presented variation methods as shown in Table. 1. were considering Δ as a constant value, i.e., independent of j and n, except for linear variation which corresponds to the continuous formulation uses $\Delta = 1/\sqrt{j}$.

The experiments to solve 3-SAT problems have shown that the performance of the algorithm remains good with constant Δ values, provided that Δ is near some

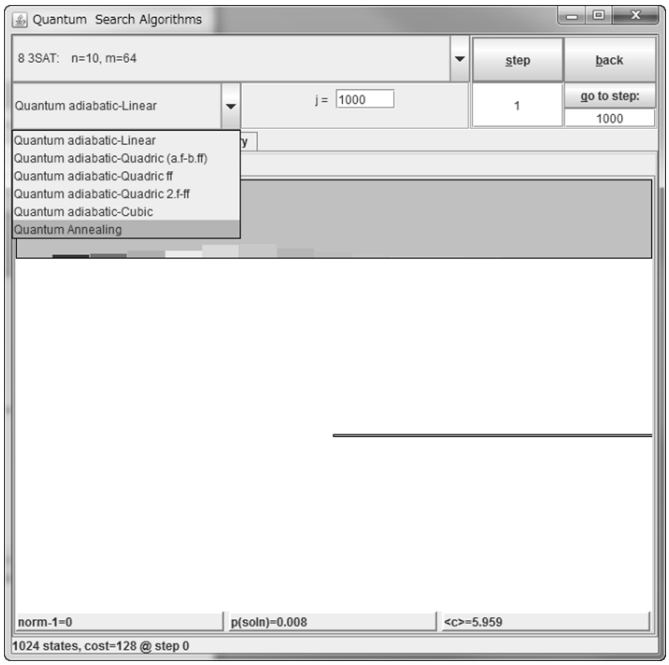


Figure 5.9: The initialization of the program where we chose the problem instance and algorithm.

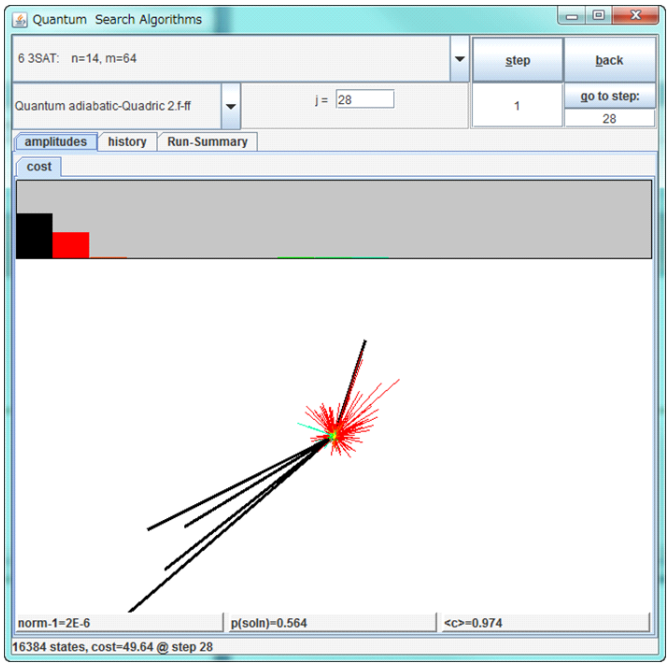


Figure 5.10: The amplitude tab of the interface shows the amplitudes as lines in a complex plane.

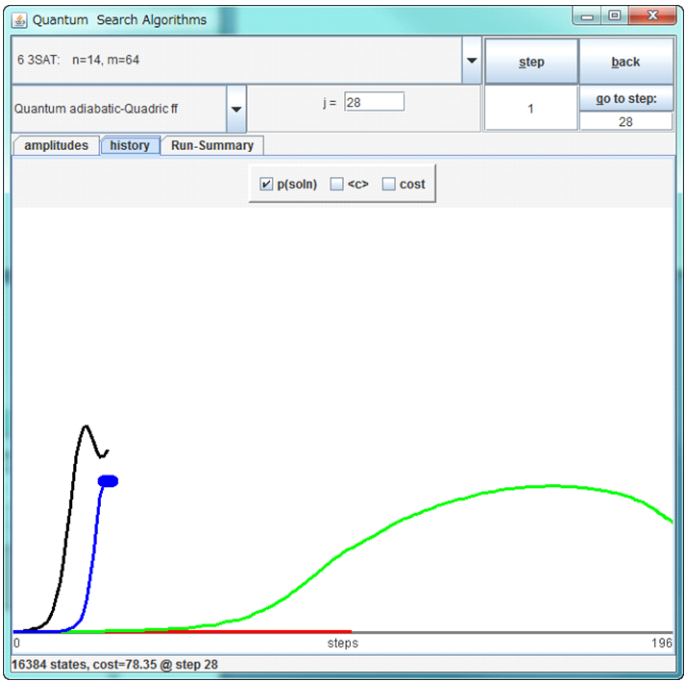


Figure 5.11: The probability tab of the interface shows propability of finding the solution *vs*. Steps for each compared algorithms.

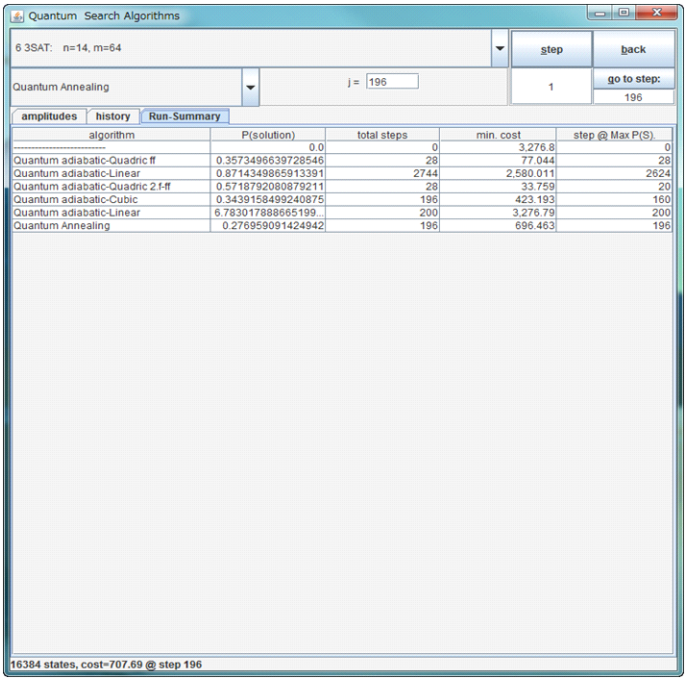


Figure 5.12: The Run-Summary tab summarizes the experimental results for each used algorithm.

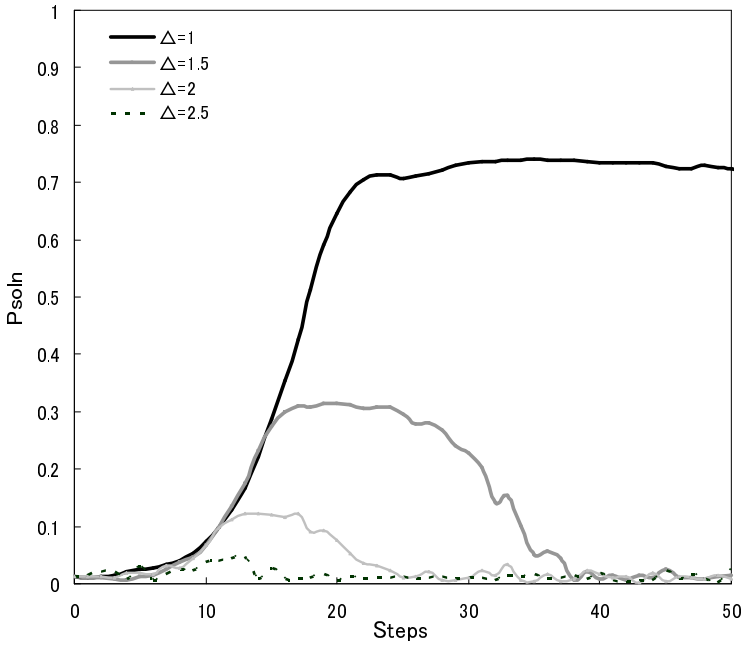


Figure 5.13: P_{soln} *vs* number of steps j for the algorithm with quadric variation solving a 8-variable 3-SAT instance with m=34 and 2 solutions, the solid black curve represents $\Delta = 1$, the solid gray curve for $\Delta = 1.5$, the thin gray curve for $\Delta = 2$, and the dashed curve for $\Delta = 2.5$.

threshold value. For problems with $n \leq 20$ this threshold found to be near 1.

Fig. 5.13 shows the algorithm behavior using the quadric variation method with different values for Δ 1, 1.5, 2, and 2.5. The figure shows that the a good performance is achieved when $\Delta = 1$ where P_{soln} goes to 0.73 with moderate number of steps $j = 30$ near ($.5 * n^2$), and as the value of Δ increases the P_{soln} goes to zero, which mean when Δ is too large (larger than 2) the initial state vector $|\psi^{(h)}\rangle$ follows the evolving eigenvector to the wrong state when f=1, giving P_{soln} goes to zero as j increases, as result of this results we use value $\Delta = 1$ for the experiments in next sections except for the linear variation which uses $\Delta = 1/\sqrt{j}$.

5.8 The Energy Gap

The energy gaps plays an important role in the quantum adiabatic evolution, where adiabatic theorem states that the evolution from the initial ground state to final

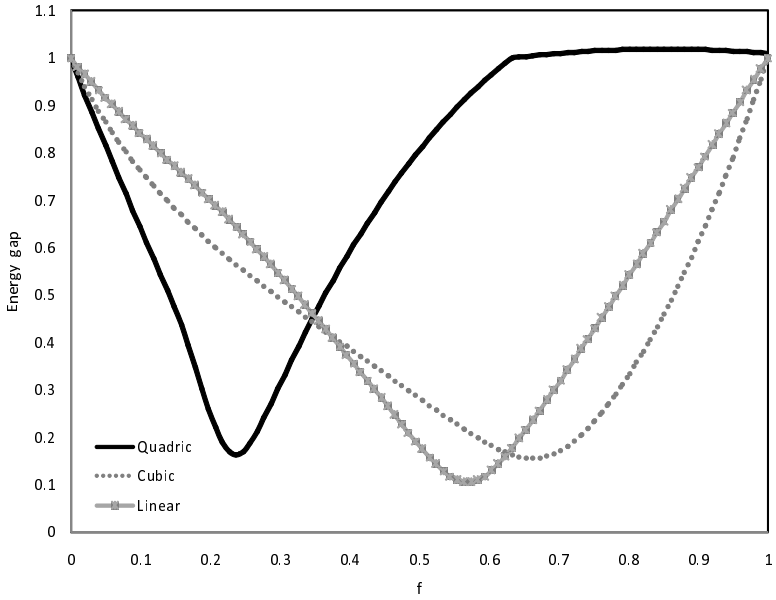


Figure 5.14: The energy gap *vs.* f for the four variation methods. The values are the difference between ground state eigenvalue and first higher eigenvalue corresponds to non-solution of the evolving Hamiltonian $H(f)$ for an instance of 3-SAT problem with n=8 and one solution.

ground state be successful provided that the evolving Hamiltonian energies (eigenvalues) never cross, i.e., with non-zero gap between the relevant eigenvalues of the system Hamiltonian $H(f)$.

The energy gap $g(f)$ is the difference between the ground state eigenvalue (the lowest eigenvalue corresponds to the solution) and the smallest higher eigenvalue corresponding to a non-solution in $H(f)$. The search cost of the adiabatic algorithm is dominated by the growth of $1/G^2$, where $G = min_f g(f)$ is the minimum gap over f, i.e., the method with the bigger G has a better chance of decreasing the search cost of the algorithm. Fig. 5.14 shows G for the three methods linear, cubic, and quadric, as 0.10257, 0.1529, and 0.18217, respectively. From these values we can see that the quadric method gives bigger value for G compared with the other methods, giving insights for decreasing the average search cost.

Table 5.1: The variation methods and the parameter configuration.

Variation methods	$\rho(f)$	$\tau(f)$	Number of steps j	Δ
Linear	f	$1-\rho(f)$	n^3	$1/\sqrt{n^3}$
Quadric	$a \cdot f - b \cdot f^2$	$1-\rho(f)$	$n^2/2$	1
Cubic	$1.921f - 2.665f^2 + 1.782f^3$	$1-\rho(f)$	n^2	1

5.9 The Experimental Results and Discussions

In this section we presents the experimental results from solving random instances of 3-SAT problems using the proposed method and the methods we described in previous sections linear and cubic. As results, the corresponding probability of finding solution and the search costs are shown.

5.9.1 The Parameter Configurations

Table 5.1 shows the different variation methods for the phase shift $\rho(f)$, and mixing functions $\tau(f)$, and values of the parameter Δ used for each method as well as the required number of steps j in the experiments. Although, the parameter Δ is originally to parametrize the time to perform the steps (operators) of the adiabatic algorithm. Here we use the constant value of the parameter Δ, i.e., independent of n and j, which proved to give a good results with cubic and other variation methods [68] [84]. We use $\Delta = 1$ for cubic method rather using it's original value 1.3218 as originally presented in [68], where $\Delta = 1$ proved to give better results as we will show in next sections.

5.9.2 Algorithm Search Behavior

In this section we compare the search behavior of the discrete adiabatic algorithm using the three different variation methods as summarized in Table.1. First is the linear variation method corresponds to the continuous adiabatic algorithm, this method uses $\Delta = 1/\sqrt{j}$, phase shift $\rho(f)$ and mixing $\tau(f)$ functions varies monotonically as linear function of f, and number of steps j grows as cubic number of bits

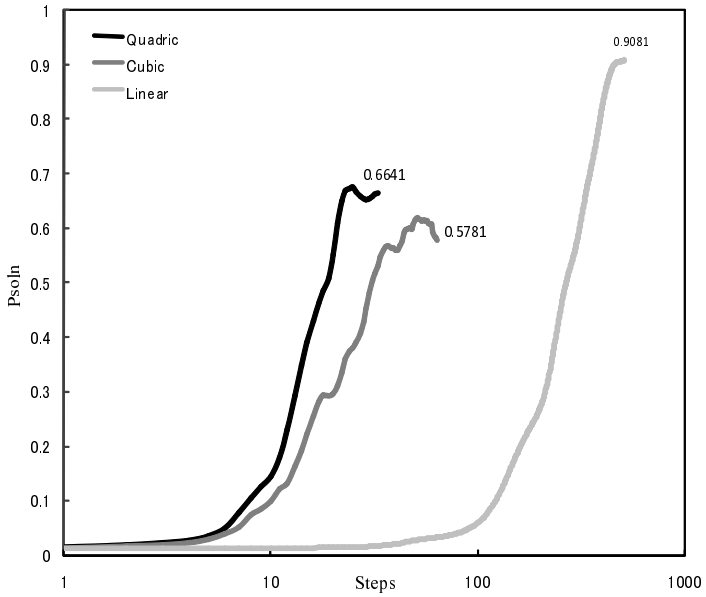


Figure 5.15: P_{soln} *vs* number of steps h for the 3 methods averaged over 10 random instances of 3-SAT problems with $n = 8$.

n^3. Second method is the cubic variation with $\rho(f)$ and $\tau(f)$ varies monotonically as cubic polynomial in f, constant Δ, and uses j grows as square number of bits n^2, and finally the quadric variation method as described in previous section.

Fig. 5.15 compares the search behavior of discrete adiabatic algorithm in solving random instances of 3-SAT problem using quadric, cubic, and linear variation methods for $n = 8$. The fig shows that the quadric variation method requires number of steps only near $n^2/2$, i.e., only 32 steps, to achieve moderate $P_{soln} \approx 0.62$, which gives faster search behavior when it is compared with the algorithm behavior using cubic variation which needs number of steps at least as n^2, i.e, 64 Steps, to achieve $P_{soln} \approx 0.51$. The Figure also shows that the linear variation corresponds to the continuous version of algorithm gives high P_{soln} in most of the trials near 0.91 using $\Delta = 1/\sqrt{n^3}$. However, it requires number of steps j at least grows as n^3 which results in high search cost. The results are the average of solving 50 random instances of 3-SAT problems.

Fig. 5.16,and Fig. 5.17 also compare the search behavior of discrete adiabatic algorithm in solving random instances of 3-SAT problem using quadric, cubic, and

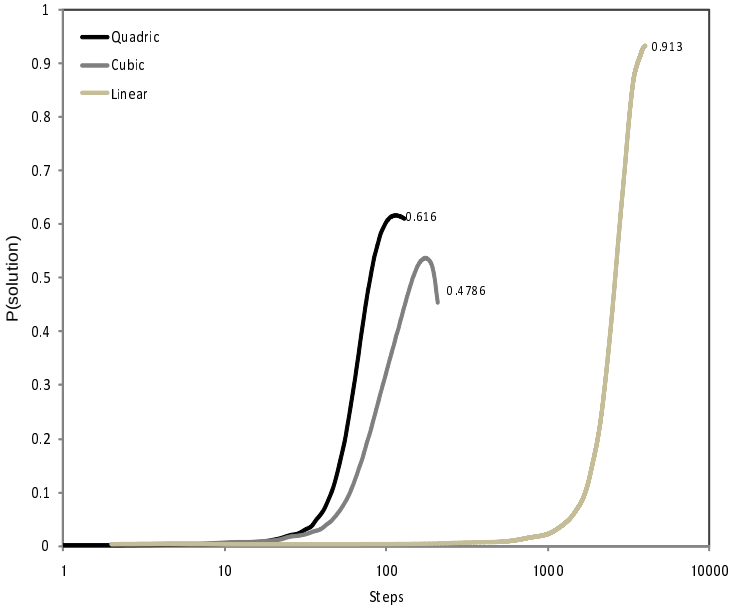


Figure 5.16: P_{soln} *vs* number of steps h for the 3 methods averaged over 10 random instances of 3-SAT problems with $n = 16$.

linear variation methods for n as 16, and 20 respectively. The figures show that the quadric variation method continue to show the same search behavior for $n = 8$ and $n = 16$ where it required only 128 and 200 steps, respectively, i.e., $n^2/2$ to achieve moderate $P_{soln} \approx 0.62$. By contrast the results of cubic variation method start to show decrease in the achieved P_{soln} as ≈ 0.421 even with using the same required number of steps as n^2. The algorithm behavior with linear variation method continue to show consistent results where it continue to achieve high $P_{soln} \approx 0.91$ with number of steps grows as n^3.

Fig. 5.18 summarizes the experimental results showing the median P_{soln} *vs*. the number of bits n for the algorithm using the three methods. This figure shows that the quadric variation method with only j grows as $.5 * n^2$ gives moderate P_{soln} near 0.62 continue at least for $n \leq 20$. Therefore the median search costs are $O(n^2/2)$, giving substantial improvement over all known classical method if it continue for larger n. Also it shows cubic variation with j grows as n^2 start with fairly high P_{soln} with moderate value near 0.5 at small $n = 8$, However, it shows decrease in the achieved P_{soln} as number of bit n increases but still gives search cost of

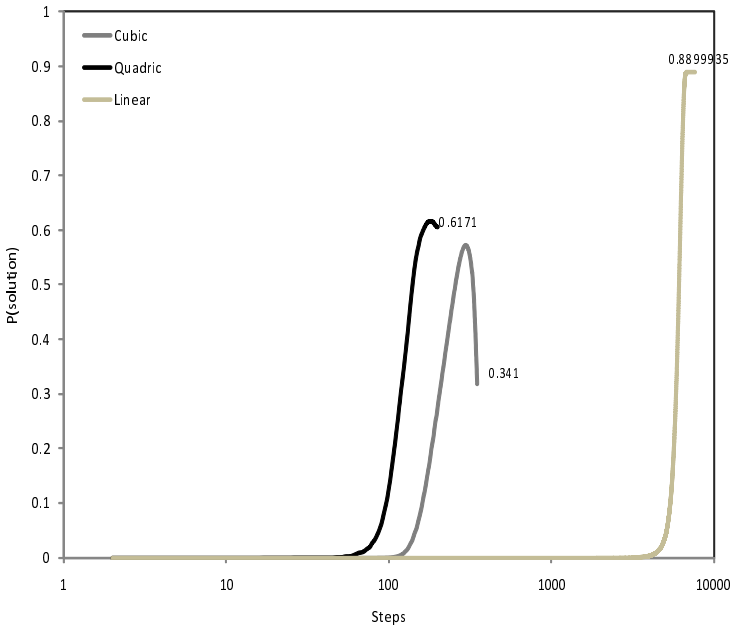


Figure 5.17: P_{soln} *vs* number of steps h for the 3 methods averaged over 10 random instances of 3-SAT problems with $n = 20$.

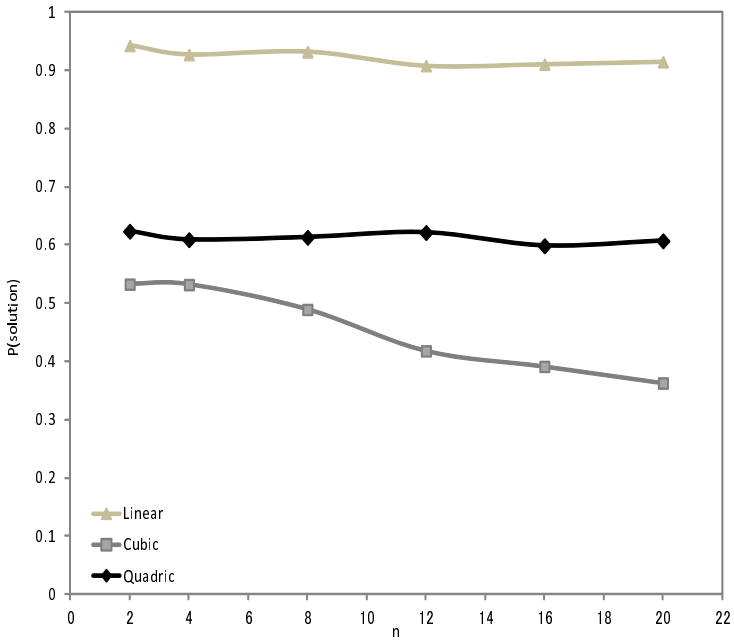


Figure 5.18: The median of P_{soln} for the algorithm *vs*. the number of bits n using the three methods. The same instances were solved using each method. we see 50 instances each n.

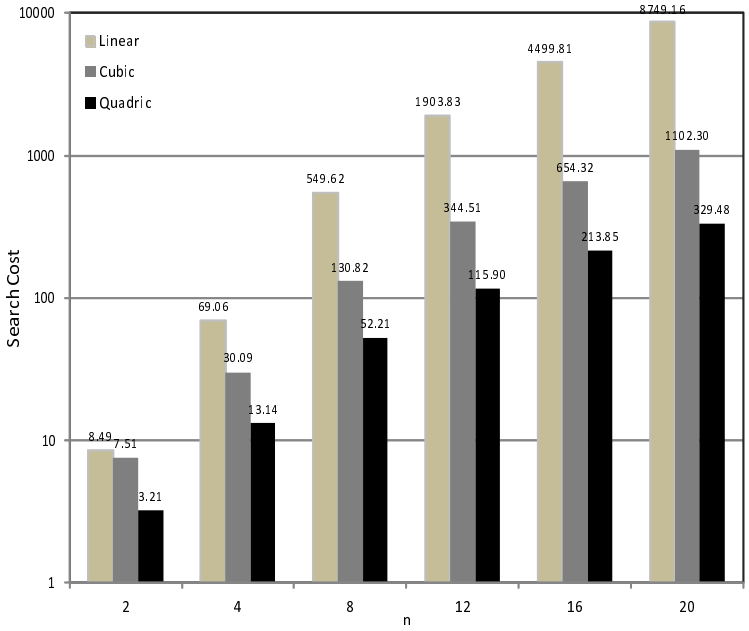


Figure 5.19: Log plot of the median search cost vs the number of variables n for the three variation methods. The values are based on the same instances of 3-SAT problems used in Fig. 5.18

$O(n^2)$. Although, the linear variation method gives P_{soln} near to 1 for all $n \leq 20$, it requires j of $O(n^3)$ which results in increasing search cost C compared with other methods. For comparison at $n = 20$, P_{soln} are 0.6327, 0.3187, and 0.9143 for quadric, cubic, and linear, respectively. This results have revealed the improvement for the search behavior of the discrete adiabatic algorithm using the quadric variation when compared with the other methods linear and cubic. Also non-stable behavior shown by the cubic variation method has raised the need of finding the better values of parameter Δ to improve the performance of the algorithm as n increases which we will present in the next chapter.

The experiments also have shown that using constant Δ, i.e., independent on n and j could give fairly high P_{soln} values, showing better use of quantum coherence in the discrete formulation of adiabatic algorithm.

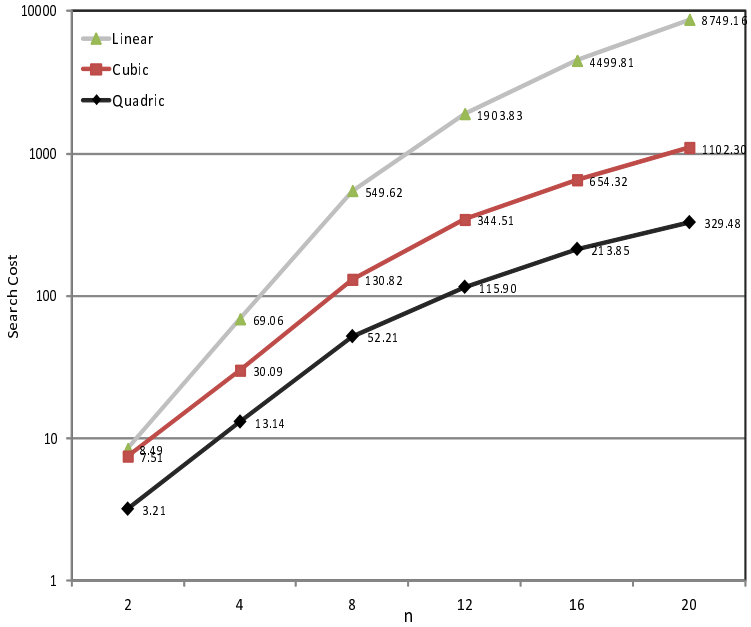


Figure 5.20: Log plot of the median search cost *vs* the number of variables n using the lines to show the grouth rate. The values are for the same instances of used in Fig. 5.18

5.9.3 Resulting Search cost

The search cost is defined to be the expected number of steps required to find a solution $C = j/P_{soln}$ providing a commonly used proxy for the computational cost of discreet methods. We used the median search cost instead of the average cost to indicate the typical behavior of the algorithm.

Fig. 5.19 compares the median search cost C of the three variation methods when used with the adiabatic algorithm. The figure shows that using quadric variation with just enough number of steps $j = 0.5 * n^2$ to achieve moderate P_{soln} as shown in Figurs. 5.15, 5.16, and 5.17 reduces the search cost when compared with the other methods. However, the algorithm with linear variation could achieve P_{soln} near 1 in most of the trials as shown, the number of steps required grows as n^3 giving a large search costs far higher than those of other methods. The experiments show that also the cubic variation could reduce the search cost compared with linear variation due to the number of steps only grows as $j = n^2$, However, the loss of Constant behavior as n increases lead to exponential growth of the search cost. For comparison the

figure shows that the median search cost at $n = 20$ for the quadric, cubic, and linear methods to be 329, 1109, and 8749, respectively. which shows improve in the resulting cost reduction using quadric variation method, due to its fairly high values for P_{soln} and reduced number of steps.

Fig. 5.20 Shows the same results shown in Fig. 5.19 but in lines chart type in order to compare the growth rates of each method. This fig shows the exponential fit to the adiabatic algorithm for the linear , cubic , and quadric are $e^{1.132 \cdot n}$, $e^{1.02 \cdot n}$, $0.5 \cdot e^{0.932 \cdot n}$, respectively . This fits show the improvement in the growth rate for the quadric variation, However, it is a bit small improvement compared with the cubic variation.

Here, the search cost is defined to be the expected number of steps required to find a solution $C = j/P_{soln}$ which is a commonly used proxy for the computational cost of the discrete methods [68], pending further study of the clock rates of the used operators.

5.10 Conclusions

Quantum adiabatic algorithm is a method of solving computational problems by evolving the ground state with a slowly varying Hamiltonian. This algorithm is a remarkable discovery because it offers new insights into the potential usefulness of quantum resources in solving combinatorial search problems.

In this chapter, we have presented a new partial monotonic variation method for the phase function in the discrete adiabatic algorithm in attempt to improve the search behavior and decrease the over all search cost. this method is named quadric variation method where it presents the phase functions $\rho(f)$ and $\tau(f)$ as a quadric polynomial in the step function of the algorithm f. We have studied the minimum energy gap for the proposed method to examine if it obey the adiabaticity condition or not and to compare with other methods. The experiments are carried out using the discrete formulation of the adiabatic algorithm in to solve random instances of 3-SAT problems to compare the proposed method and the recently proposed methods such as linear variation , cubic variation method. With problem sizes feasible to

evaluate by using simulation on current computers. The experiments have revealed that using the quadric variation method improves on other variation methods cubic and linear, in resulting search cost and search behavior on average. And also it shows that this method could present farther improvement with better values of a, b, and the parameter Δ which will present in the next chapter.

Chapter 6

Study on speeding up the quantum adiabatic computation in satisfiability problems

6.1 Introduction

In this chapter we extend the experimental study presented in the previous chapter 5 concerning the discrete version of quantum adiabatic algorithm [92]. Here we present a complete monotonic version of the quadric variation method for the phase functions of the adiabatic algorithm. We aim to speed up the algorithm and decrease the overall search cost on average. In addition, we present improved values for a parameter Δ in the algorithm to be used with quadric variation as well as the previously proposed methods. Experiments are carried out to examine the performance of the discrete adiabatic algorithm by solving satisfiability problems (3-SAT) using different variation methods such as, linear, cubic, annealing and the proposed method. Two sets of parameter values are considered in the experiments, the original and the improved values. As a result, the corresponding probability of finding the solution, energy gaps and the search costs are shown.

6.2 The Discrete adiabatic algorithm

Originally, the adiabatic method is continuous process. However, in this paper we use the algorithmically equivalent discrete formulation of the algorithm [68] [84] acting on the amplitude state vector initially in the ground state of the Hamil-

tonian $H^{(0)}$, which can be represented for n qubits quantum system as $|\psi_s^{(0)}\rangle = \frac{1}{\sqrt{2^n}}[1, 1, ..., 1]^T$ [68]. Discretization of a continuous Hamiltonian is straightforward process and proved to change the total run time of the algorithm only polynomially [70].

Consider a discreet Hamiltonian $H(f)$ in the general form

$$H(f) = \tau(f)H^{(0)} + \rho(f)H^{(c)}, \tag{6.1}$$

where $\tau(f)$ and $\rho(f)$ are mixing and phase shift function, respectively. Both of them are functions of f $(0 \leq f \leq 1)$, shown in Table.1, subject to the boundary conditions

$$\tau(0) = 1, \rho(0) = 0, \tag{6.2}$$

$$\tau(1) = 0, \rho(1) = 1. \tag{6.3}$$

Although, the two functions $\tau(f)$, and $\rho(f)$ are not necessary to be monotonic [85] (i.e. obey the constraint $\tau(f) + \rho(f) = 1$), here we consider only the monotonic functions. In matrix form [68], the Hamiltonian $H^{(c)}$ could be presented as

$$H_{r,s}^{(c)} = c(s)\delta_{r,s}, \text{where} \quad \delta_{r,s} = \begin{cases} 1 & \text{if } r{=}s \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

This Hamiltonian introduces a phase shift factor in the amplitude of assignment s depending on its associated cost $c(s)$, where the higher cost results in more phase shift. The Hamiltonian $H^{(0)}$ can be implemented with elementary quantum gates by use of the Walsh-Hadamard transform with elements $W_{r.s} = 2^{-n/2}(-1)^{r \cdot s}$ [68], where $H^{(0)} = WDW$ and D is a diagonal matrix with the value for state r given by the sum of the bits, i.e, the element $D_{r,r}$ is just a count for the number of bits equal to 1 in state r.

A single trial of the algorithm, as shown in Fig. 6.1, consists of j steps with parameter Δ and can be described as follows [69]:

1. Initialize the amplitude state vector to the ground state of $H^{(0)}$ giving equal values for all states as $|\psi_s^{(0)}\rangle = \frac{1}{\sqrt{N}}[1, 1, ..., 1]^T$.

2. For steps $h = 1$ to j, repeat the matrix multiplication :

$$|\psi^{(h)}\rangle = U_h(f)|\psi^{(h-1)}\rangle. \tag{6.5}$$

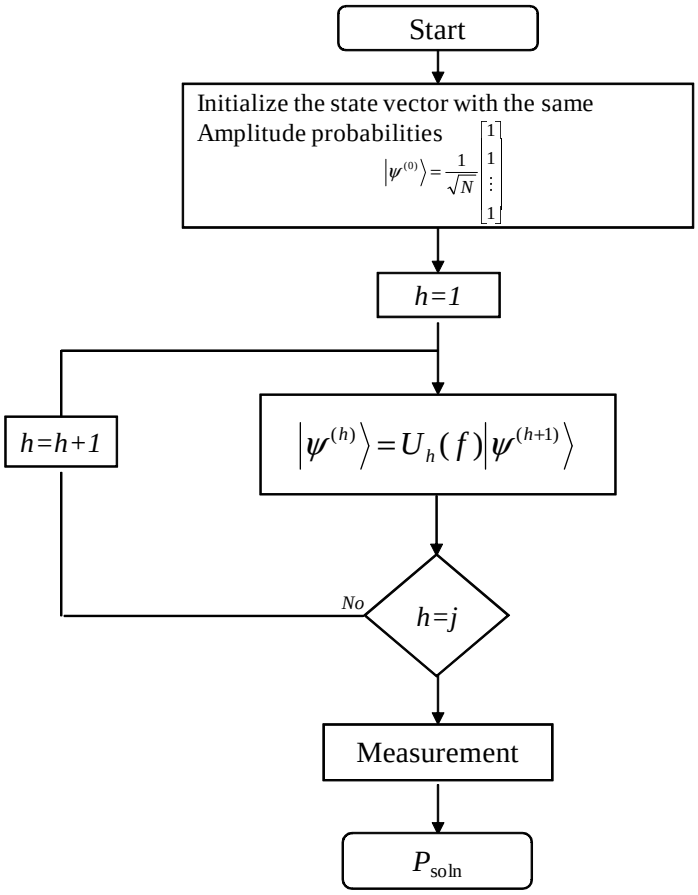


Figure 6.1: Flowchart represents the discrete version of adiabatic quantum algorithm.

where $|\psi^{(h-1)}\rangle$ is the amplitude state vector at the step $h-1$, and $U_h(f)$ is unitary evolution operator for h^{th} step which can be represented as

$$U_h(f) = e^{-i\tau(f)H^{(0)}\Delta} \cdot e^{-i\rho(f)H^{(c)}\Delta}. \tag{6.6}$$

3. Measure the final system after the j steps take place. Then probability to find a solution is given by $P_{soln} = \sum_s ||\psi^{(j)}||^2$.

6.3 The Monotonic Quadric variation method

Recently, various variation methods (scheduling methods as named in other articles) for the phase shift function $\rho(f)$ and the mixing function $\tau(f)$ in the adiabatic algorithm have been presented in order to speed up the algorithm and decrease the

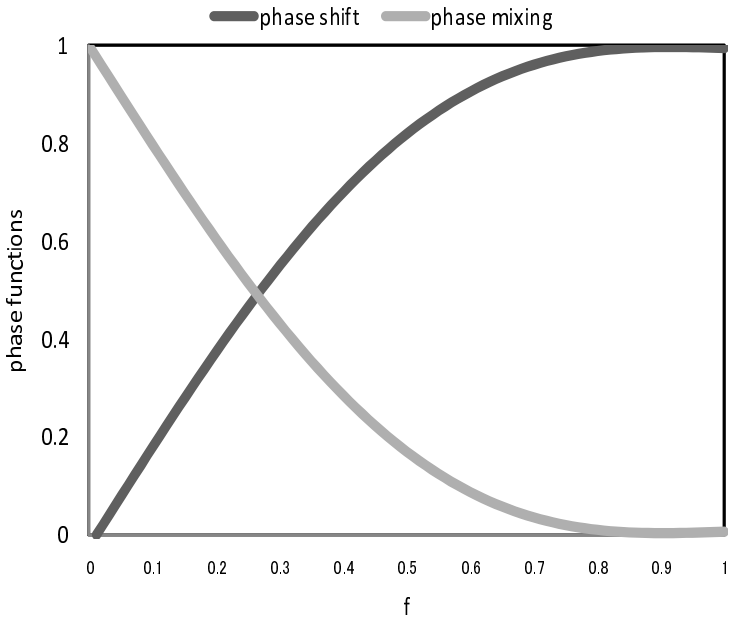


Figure 6.2: Phase shift and mixing functions *vs.* f for the Quadric variation method.

overall search cost. Some of these methods was monotonic (means $\rho(f)+\tau(f)=1$ for all values of f) such as linear variation [24], cubic variation [68], and the scheduling method in quantum annealing [14], and others were non-monotonic such as the quadric formula has been used in [85] to solve unstructured search problem, in that article $\rho(f) = f + A \cdot f(1-f)$ and $\tau(f) = 1 - f + A \cdot f(1-f)$ are used where A is a constant value.

Here we propose a complete monotonic version of the quadric variation method for $\rho(f)$ and $\tau(f)$, where $\rho(f) = 2 \cdot f - f^2$ and $\tau(f) = 1 - \rho(f)$. This formula are easy to construct, and it was found to satisfy the adiabaticity conditions, i.e, using this method doesn't lead the relevant eigenvalues of the evolving Hamiltonian $H(f)$ to cross. This formula is the improved version of what we present in the previous chapter where it was partially monotonic. We aim from using this formula to speed up the search behavior of the adiabatic algorithm.

Fig. 6.3 shows the Lowest 3 eigenvalues of evolving Hamiltonian $H(f)$ *vs.* f in solving an instance of 3-SAT problem with one solution. In this evolution the quadric variation method is used for the phase shift $\rho(f)$, and mixing $\tau(f)$ functions.

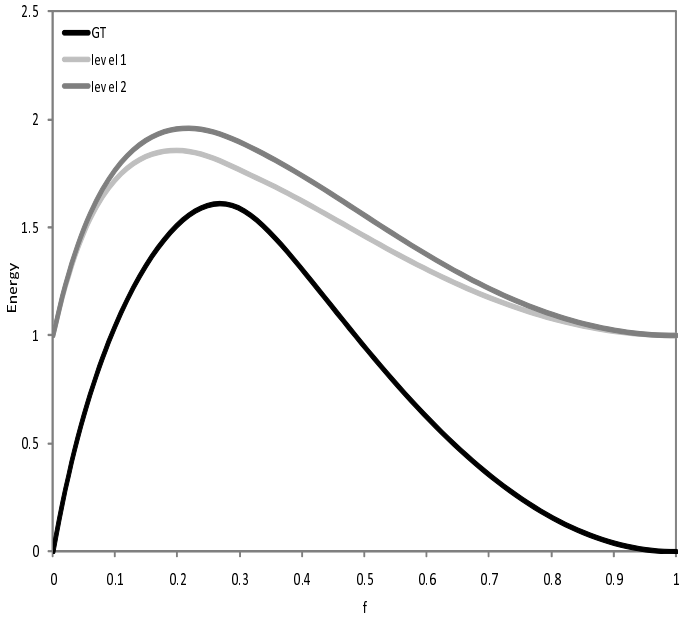


Figure 6.3: Lowest two 3 eigenvalues vs. f for an instance with n=8, and one solution. This instance was solved using the algorithm with quadric variation method. The black curve shows the ground state, gray curve and dark gray curve represent the smallest higher eigenvalues (energies correspond to non solutions) level 1, level 2, respectively.

This figure shows that the eigenvalue of the ground state (bold black curve) and the next two smallest higher eigenvalues correspond to non solutions (gray and dark gray curve respectively), never cross, i.e., with non-zero gap. Simply, this figure reveal that evolution under the discrete Hamiltonian $H(f)$ with the quadric variation method follows the adiabatic theorem.

6.4 Numerical Example

In this section we present an example for the algorithm search behavior within few steps in order to show how the values of the probability amplitudes shifts for each step h using Quadric variation method. The amplitudes values are initialized with equal values (Normalized) for all possible states. The amplitudes are presented in a complex plane with aim to show the shift in the phases and the value mixing during the steps. In this complex plane X-axis presents the real part of the amplitude and Y-axis presents the imaginary part. (The amplitude is a complex number in the

Table 6.1: an Example for 1-SAT problem.

Assignment	Number of Conflicts c	Corresponding Integer
00	0	0
01	1	1
10	1	2
11	2	3

form $a + ib$).

Consider a 1-SAT problem with two variables v_1 and v_2. Given an expression

$$(NOT(v_1))AND(NOT(v_2)) \tag{6.7}$$

This expression (an instance) has unique solution $|00\rangle$ among four possible states as shown in the Table 6.1. Conflict number is the number of unsatisfied clauses for the assignments.

For initialization of the algorithm we prepare the following

1. Initialize the state vector with equal values of the amplitudes for all possible states as $1/\sqrt{N}$, where $N = 2^n$ and n is the number of variables(Qubits) considered. so the initial state vector $|\psi^{(0)}\rangle$ is the uniform superposition of all possible states (here the all possible state are 4) as follows

$$|\psi^{(0)}\rangle = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \tag{6.8}$$

2. Prepare the initial Hamiltonian $H^{(0)}$ and the final Hamiltonian $H^{(c)}$ which encode the solution (the state with cost =0)of the problem in the lowest energy state. These Hamiltonian are presented as described in the section 5.3. For this example will be as follows

$$H^{(0)} = \begin{bmatrix} 1 & -0.5 & -0.5 & 0 \\ -0.5 & 1 & 0 & -0.5 \\ -0.5 & 0 & 1 & -0.5 \\ 0 & -0.5 & -0.5 & 1 \end{bmatrix} \tag{6.9}$$

$$H^{(c)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \tag{6.10}$$

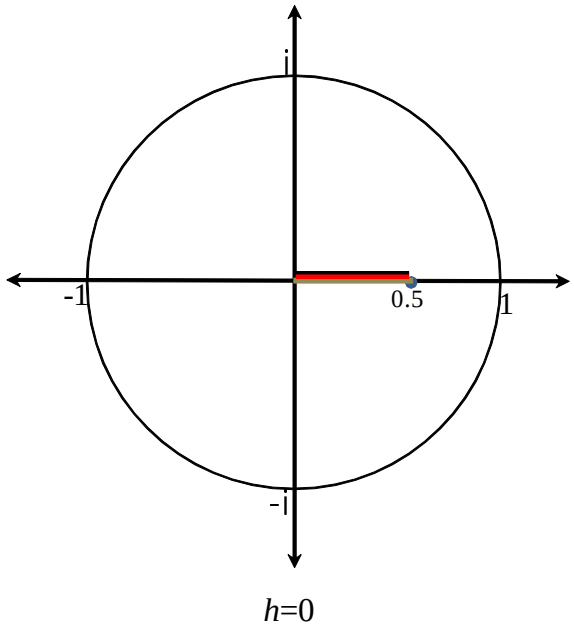


Figure 6.4: A complex plane presents The probability amplitude at the initialization step $h = 0$.

Now, Apply the unitary transformation (operator) $U_h(f)$ on the initial state vector $|\psi^{(0)}\rangle$ for j steps, i.e., for $h = 0$ to j. Here we consider 4 steps..

Figures 6.4-6.8 presents the steps of the algorithm, $h = 0$, $h = 1$, $h = 2$, $h = 3$, and $h = 4$, respectively. Each figure shows the phase shift effect on the probability amplitude associated with each state as well as the state mixing effects. The bold black curve in the figures present the solution state $|00\rangle$.
Finally, after 4 steps take place, measuring the final state vector $|\psi^{(4)}\rangle$ will give the probability of finding the solution with $P_{(soln)} = 0.729$ which is fairly high.

6.5 The parameter Δ

A good performance of the discrete adiabatic algorithm requires an appropriate choice of parameter Δ. In the discrete formulation, Δ parameterizes the unitary transform rather than determining the time T required to perform it [68]. By contrast, Δ in the continuous formulation determines the time to perform the operation as $T = j \cdot \Delta$.

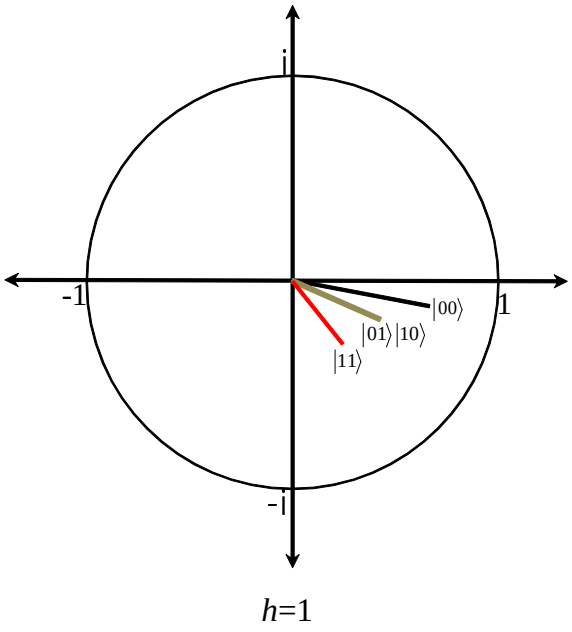


Figure 6.5: A complex plane presents The probability amplitude of all the possible states at Step $h = 1$ presented.

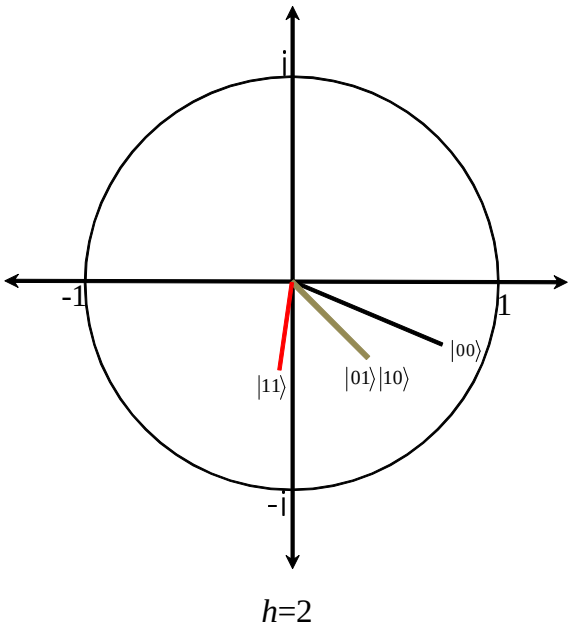


Figure 6.6: A complex plane presents The probability amplitude of all the possible states at Step $h = 2$ presented.

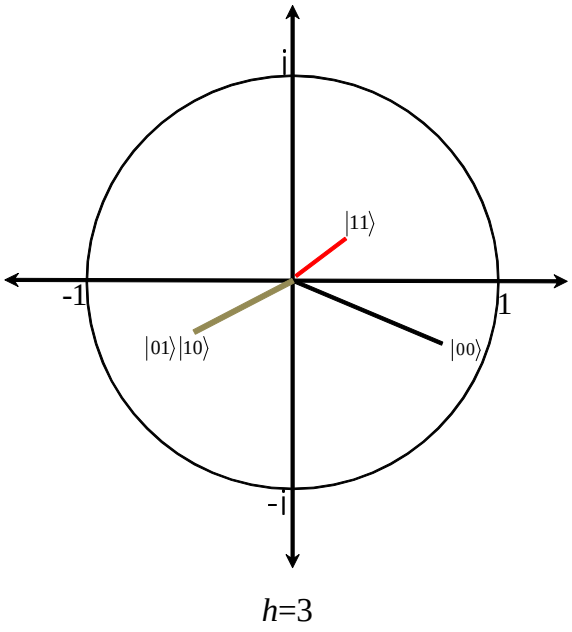


Figure 6.7: A complex plane presents The probability amplitude of all the possible states at Step $h = 3$ presented.

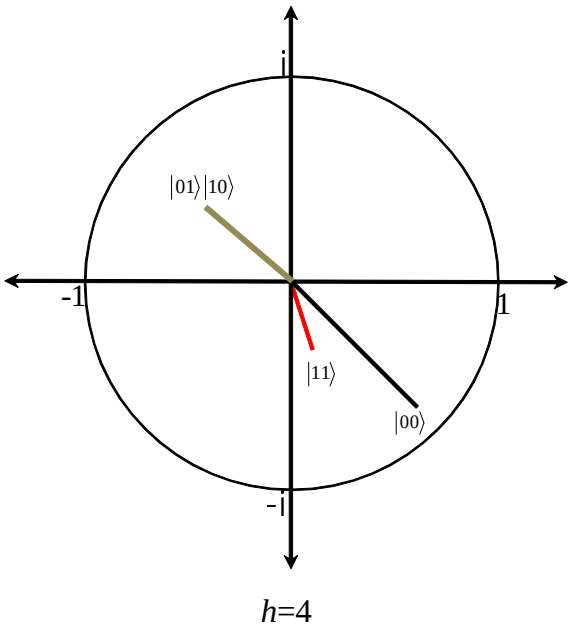


Figure 6.8: A complex plane presents The probability amplitude of all the possible states at Step $h = 4$ presented.

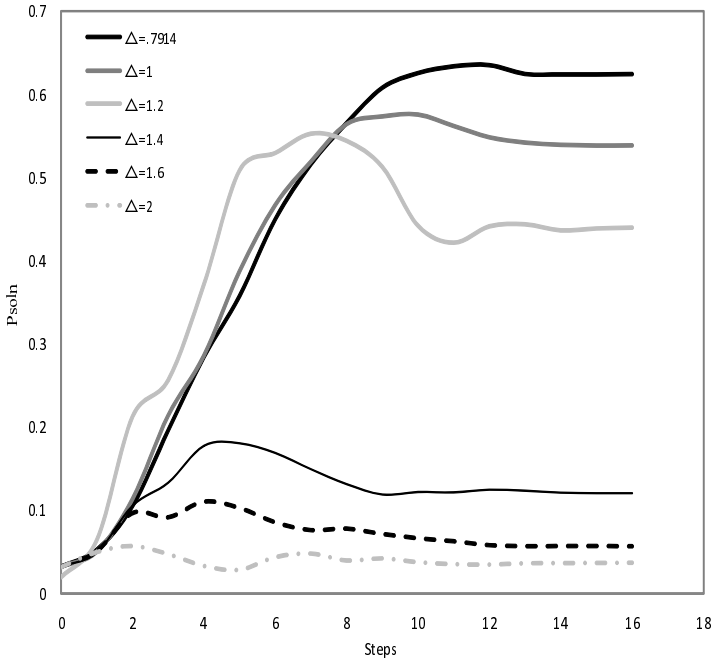


Figure 6.9: P_{soln} vs number of steps j for discrete adiabatic algorithm with quadric variation solving a 8-variable 3-SAT instance with m=34 and 2 solutions, the solid black curve represents the algorithm for $\Delta = 1$, the solid gray curve for $\Delta = 1.5$, the thin gray curve for $\Delta = 2$, and the dashed curve for $\Delta = 2.5$.

The experiments to solve 3-SAT problems presented in the previous chapter have shown that the performance of the algorithm remains good with constant Δ values, provided that Δ is below some threshold value. For 3-SAT problems with $n \leq 14$ this threshold found to be below 1. However, The performance found to decline with constant Δ for bigger n ($n > 14$). Therefore, we present the improved values of Δ in order to improve the performance of the algorithm as n increases as well as decreasing the resulting search cost.

6.5.1 The original setting of Δ

The most recently presented variation methods such as cubic variation [68], quantum annealing [14], and the first version of the quadric variation [69] were considering Δ as a constant value, i.e., independent of j and n, except for linear variation which corresponds to the continuous formulation uses $\Delta = 1/\sqrt{j}$.

Fig. 6.9 shows effects of Δ values on the performance of algorithm using the

quadric variation method, it uses different values for Δ. The figure shows that the best performance is found when $\Delta = 0.7914$ where P_{soln} could achieve 0.63 with only $j = 16$ steps. Also it shows that as the value of Δ increases over 1 the P_{soln} goes to zero, which mean when Δ is too large (larger than 2) the initial state vector $|\psi^{(h)}\rangle$ follows the evolving eigenvector to the wrong state at f=1, giving P_{soln} goes to zero as j increases.

6.5.2 The Improved setting of Δ

The experiments to solve 3-SAT problems presented in the previous chapter have shown that the performance of the algorithm remains good with constant Δ values below some threshold value. For 3-SAT problems with $n \leq 14$ this threshold found to be below 1. However, The performance found to decline with constant Δ for bigger n ($n > 14$). Therefore, which raised the need for improved values of Δ in order to maintain the consistency of the algorithm as n increases.

Table 6.2. Shows the values of the parameter Δ for each method in two sets. First, are the original values have been used in the original articles [24] [68] [14] as constant values independent of j and n. Second is the improved set which we present in order to improve the performance of algorithm with the previous methods as well as the proposed method. In the proposed formulas the parameter Δ is a constant depends on n. Specifically, Δ is chosen to raise exponentially with respect to the number of bits n. Using this formula could satisfy the condition $\Delta \to 0$ as $n \to \infty$, so the algorithm can approximate the continuous evolution, then it is expected to give higher P_{soln} as j increases. This exponential formulas are found to be the median of best Δ for each n, where 200 instances are considered for each n.

6.6 The Variation Methods

Recently several variation methods for phase shift $\rho(f)$, and mixing function $\tau(f)$ in the discrete adiabatic quantum algorithm such as linear variation [24], and cubic variation [68], and the quantum annealing [14] as shown in Table. 6.2.

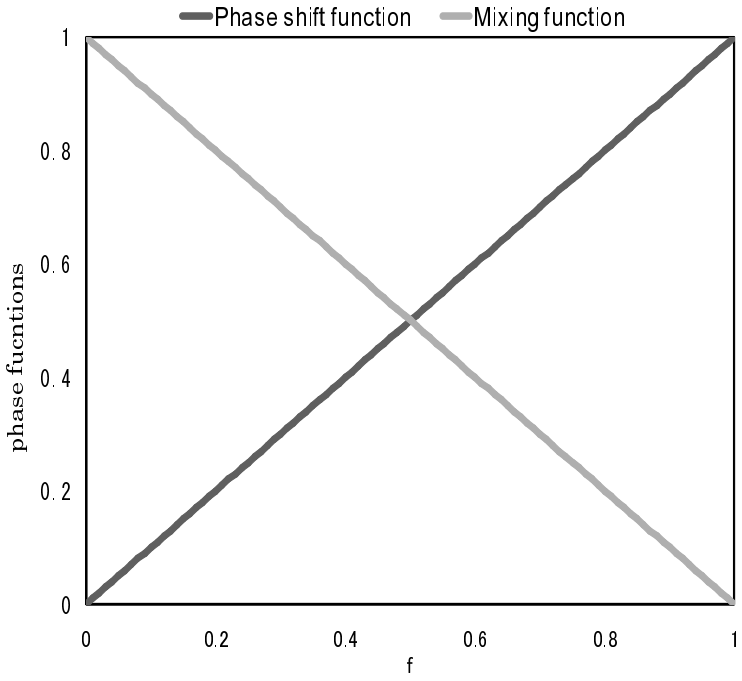


Figure 6.10: Phase shift and phase mix functions *vs.* f for the linear variation method.

6.6.1 Linear Variation Method

Linear variation method is corresponding method to the continuous version of the adiabatic algorithm (The original scheduling method for the adiabatic algorithm). It varies the phase shift $\rho(f)$ and mixing functions $\tau(f)$ of the adiabatic algorithm linearly as simple monotonic function as f and $1-f$, respectively. This method presents the original idea of the adiabatic evolution model to do a computations [24] [25]. This method is completely monotonic where it starfishes the condition ($\rho(f)+\tau(f)=1$) for all values of f and have values always bounded between 0 and 1.

Fig. 6.10 shows the phase functions for linear variation method *vs.* the f value from 0 to 1. From this figure we can note that the linear variation method has diversity area and intensification with equal size, which gives a sign for requiring an intensive phase shifting and mixing, i.e., requires higher number of steps to achieve high probability of finding a solutions.

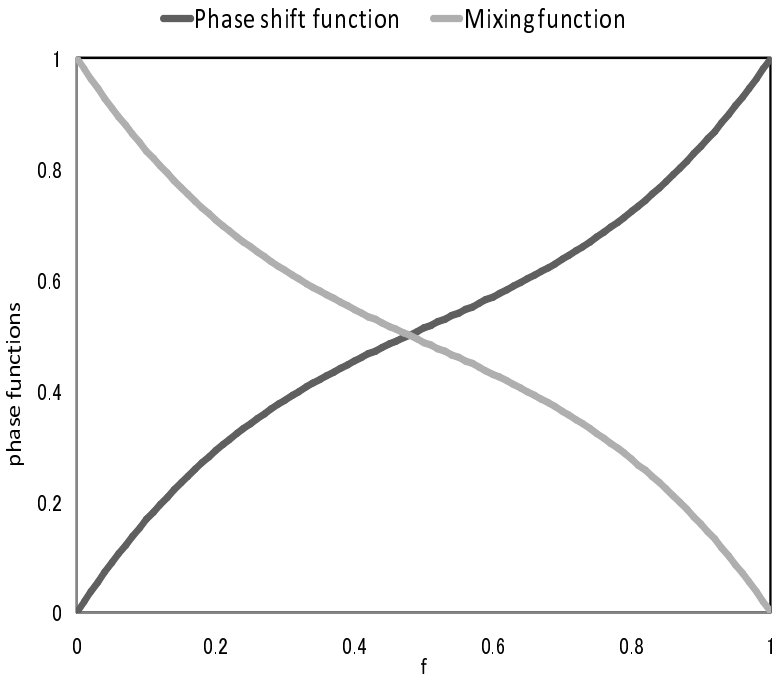


Figure 6.11: Phase shift and phase mix functions *vs.* f for the Cubic variation method.

6.6.2 Cubic Variation method

Cubic variation method is similar to the functional form optimizing the adiabatic method for unstructured search which proved to reduce the resulting cost [68] [82] [84]. In this method $\rho(f)$ and $\tau(f)$ vary monotonically as cubic polynomial in f as $1.921f - 2.665f^2 + 1.782f^3$ and $1 - [1.921f - 2.665f^2 + 1.782f^3]$, respectively. This method is completely monotonic where it satisfies the condition (ρ+τ=1) for all values of f and have values always bounded between 0 and 1. And also it satisfies adiabaticity condition, i.e., using this method never lead the relevant eigenvalues (energies) of the evolving Hamiltonian $H(f)$ to cross.

Fig. 6.11 shows the phase functions for cubic variation method *vs.* the f value from 0 to 1. From this figure we can note that the cubic variation method has diversity area and intensification with equal size (however smaller than linear variation method), which gives a sign for requiring an intensive phase shifting and mixing, i.e., it may require larger number of steps to achieve high probability of finding a solutions.

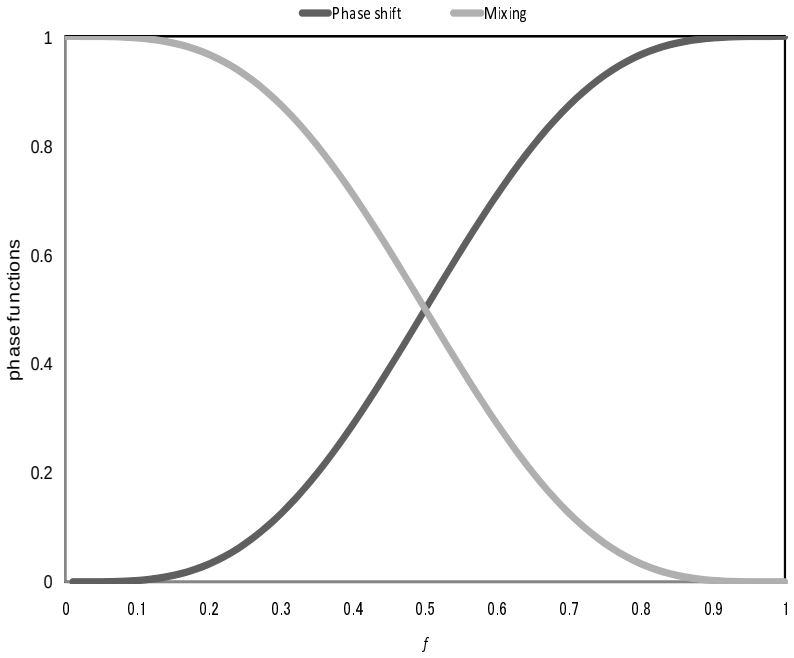


Figure 6.12: Phase shift and phase mix functions *vs.* f for the annealing method.

6.6.3 Quantum Annealing

Quantum annealing method is a scheduling method for the quantum adiabatic evolution presented in [14]. In this method $\rho(f)$ and $\tau(f)$ vary monotonically as polynomial in f as $f^4(35 - 84f + 70f^2 - 20f^3)$ and $1 - \rho(f)$ respectively. This method is completely monotonic where it satisfies the condition ($\rho(f)$+$\tau(f)$=1) for all values of f and have values always bounded between 0 and 1. And it also satisfies adiabaticity condition.

Fig. 6.12 shows the phase functions for Annealing method *vs.* f values from 0 to 1. From this figure we can see that this method has diversity area and intensification with equal size(however bigger than linear variation method).

6.7 The Minimum Energy Gap G

The adiabatic theorem states that the evolution from the initial ground state to final ground state be successful provided that the evolving Hamiltonian energies (eigenvalues) never cross, i.e., with non-zero gap between the relevant eigenvalues

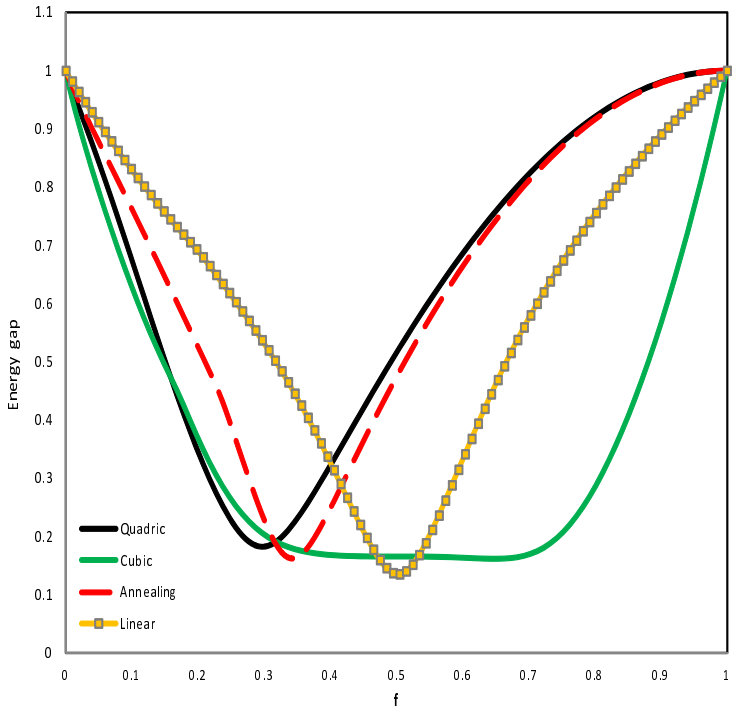


Figure 6.13: The energy gap *vs.* f for the four variation methods. The values are the difference between ground state eigenvalue and first higher eigenvalue corresponds to non-solution of the evolving Hamiltonian $H(f)$ for an instance of 3-SAT problem with n=8 and one solution.

of the system Hamiltonian H. The energy gap $g(f)$ is the difference between the ground state eigenvalue (the lowest eigenvalue corresponds to the solution) and the smallest higher eigenvalue corresponding to a non-solution in $H(f)$. The search cost of the adiabatic algorithm is dominated by the growth of $1/G^2$, where $G = min_f g(f)$ is the minimum gap over f, i.e., the method with the bigger G has a better chance of decreasing the search cost of the algorithm. Fig. 6.13 shows G for the four methods linear, cubic, annealing and quadric, as 0.1357, 0.1629, 0.16342, and 0.18217 respectively. From these values we can see that the quadric method gives bigger value for G compared with the other methods, giving insights for decreasing the average search cost.

Table 6.2: The variation methods and the parameter configuration.

Variation methods	$\rho(f)$	$\tau(f)$	Number of steps j	Δ	
				Original	Improved
Linear	f	$1-\rho(f)$	n^3	$1/\sqrt{n^3}$	$1/\sqrt{n^3}$
Quadric	$2f-f^2$	$1-\rho(f)$	$2\cdot n$	0.7914	$1.17\cdot e^{-0.021\cdot n}$
Cubic	$1.921f-2.665f^2+1.782f^3$	$1-\rho(f)$	n^2	1.3214	$1.22\cdot e^{-0.026\cdot n}$
Annealing	$f^4(35-84f+70f^2-20f^3)$	$1-\rho(f)$	n^2	1.0	$0.94\cdot e^{-0.041\cdot n}$

6.8 The Experiments and Discussions

Table. 6.2 shows the different variation methods for the phase mixing function $\tau(f)$, cost phase function $\rho(f)$, and values of the parameter Δ used in the experiments.

6.8.1 Algorithm Search Behavior

In this section we compare the search behavior of the discrete adiabatic algorithm using the variation methods summarized in Table. 6.2, i.e., linear, cubic, annealing, and quadric. Also we compare the resulting probability of finding the solution P_{soln} using the original and the improved values of the parameter Δ. The results in the next figures are the average of solving 100 random instances of 3-SAT problems, same instances are solved for comparisons.

Fig. 6.2, and Figures 6.10-6.12 show the phase functions $\rho(f)$ and $\tau(f)$ for each variation method $vs.f$ values from 0 to 1. From these figures, we can see that the quadric variation method shown in Fig. 6.3 has a smaller diversification area (the area bounded by the Y-axis and the two curves) and a bigger intensification area (the area bounded by the two curves on the right-hand side of the figure) compared with the other methods, and this indicates that the algorithm with quadric variation method should have better results.

Fig. 6.14, and Fig. 6.15 compare the search behavior of algorithm using various variation methods for number of bits $n = 16$. These figures use the original and the improved values of parameter Δ, respectively. From these figures we can see that the quadric variation method shows faster search behavior with number of steps j only as $(2\cdot n)$, i.e., only 32 steps, where it could achieve P_{soln} more than 0.5, for both

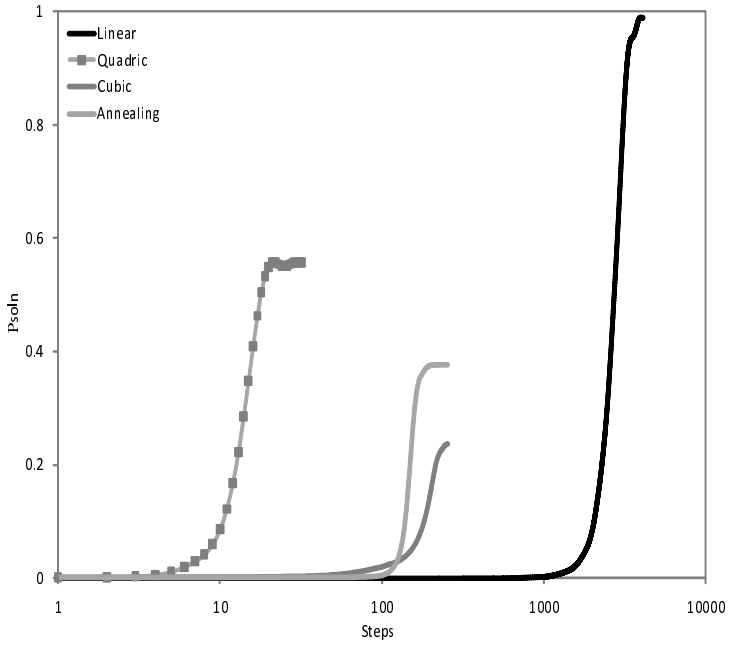


Figure 6.14: P_{soln} *vs* number of steps h for the 4 methods using original values of parameter Δ. The figure shows the average behavior calculated over 100 random instances of 3-SAT problems with $n = 16$.

values of parameter Δ, which is fairly high, giving 0.541 and 0.5874 with constant, and improved Δ values, respectively, with near 10% improve in case of using the improved value of Δ. Also these figures show cubic and annealing methods with number of steps grow as n^2 gives average P_{soln} as 0.361 and 0.224, respectively, using original values of Δ. And In case of the improved values of Δ, the P_{soln} improves to achieve 0.85 for cubic and over 0.92 for annealing method, which shows great enhancement of the search behavior of the algorithm with improved values of Δ, However, annealing and cubic methods still require number of steps j at least n^2 ,i.e., 256 steps.

Fig. 6.16, and Fig. 6.17 compare the search behavior of algorithm using various variation methods for number of bits $n = 24$, for both the original and the improved values of parameter Δ, respectively. These figures show that the algorithm with quadric variation method continues showing faster search behavior giving fairly high P_{soln} as 0.522 and 0.581 with constant and improved values of parameter Δ, respectively. Also the algorithm with cubic and annealing shows decreasing in the

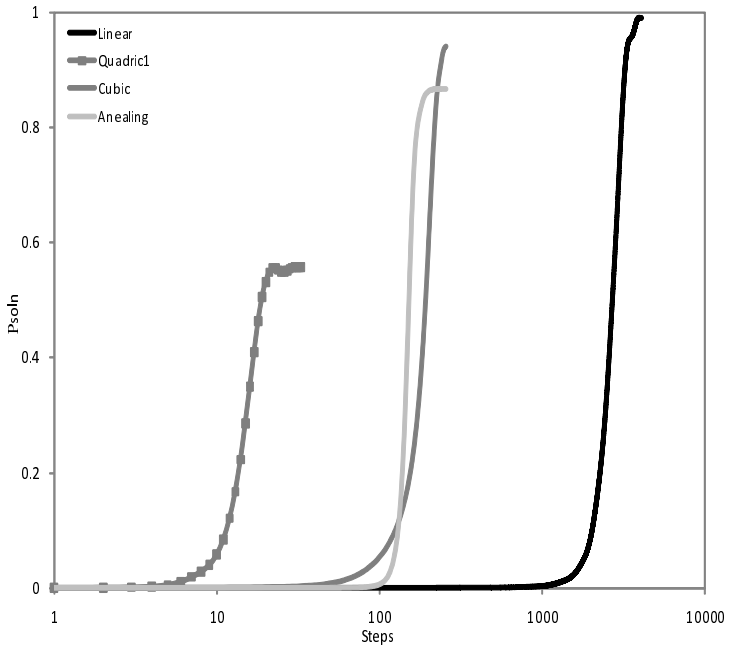


Figure 6.15: P_{soln} *vs* number of steps h for the 4 methods using improved values of parameter Δ. The figure shows the average behavior calculated over 100 random instances of 3-SAT problems with $n = 16$.

resulting P_{soln} as 0.123 and .147, respectively with original values of parameter Δ. On the other hand the algorithm using cubic and annealing methods with improved Δ values shows a great enhancement in resulting P_{soln} as 0.931 and 0.854, respectively. The linear variation corresponds to the continuous version of algorithm gives high P_{soln} in most of the trials near 95% using $\Delta = 1/\sqrt{n^3}$ for both n=16 and 24. However, it requires number of steps j at least grows as n^3 which results in high search cost. The results shown in figures 6.14,6.15,6.16, and 6.17 are the average of solving 100 random instances of 3-SAT problems.

Fig. 6.18 summarizes the experimental results showing the average P_{soln} *vs*. the number of bits n up to 28 bits using the original values of parameter Δ (constant in case of quadric variation). This figure shows that the quadric variation method with only j grows as $2 \cdot n$ gives moderate P_{soln} between 0.52 and 0.58 continue for all n up to 28 bits showing stable performance when compared with other methods where, cubic and annealing methods start at $n = 8$ giving higher P_{soln} as 0.839 and 0.862, respectively. Then the achieved P_{soln} decreases as number of bit n increases. For

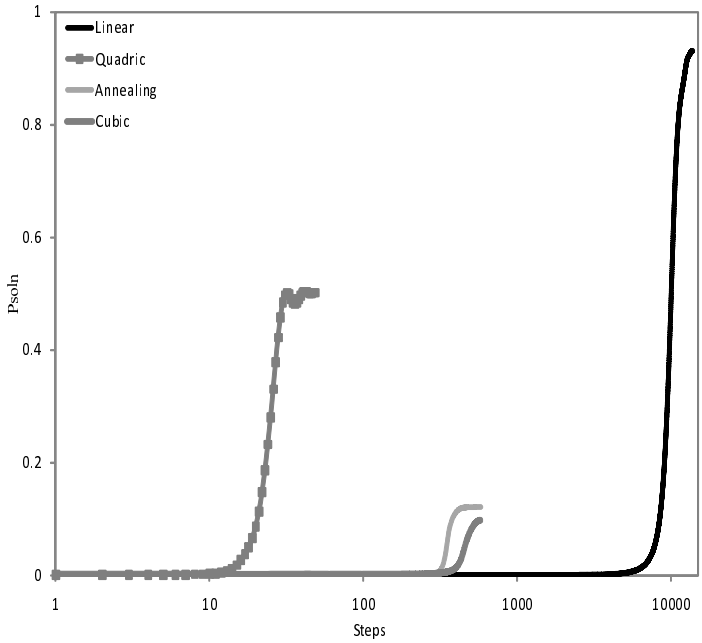


Figure 6.16: P_{soln} *vs* number of steps h for the 4 methods using original values of parameter Δ. The figure shows the average behavior calculated over 100 random instances of 3-SAT problems with $n = 24$.

comparison at $n = 28$, P_{soln} riches 0.0887 for cubic and 0.1283 for annealing. This results has raised the need of finding the better values of parameter Δ to improve the performance of the algorithm as n increases.

Fig. 6.19 summarizes the experimental results showing the average P_{soln} *vs*. the number of bits n up to 28 bits using the improved values of parameter Δ. This figure shows that the quadric variation method continue to show stable performance for n up to 28 bits, However with near 10% improvement in the resulting P_{soln} to be between 0.556 and 0.601. Also using the improved values of Δ as exponential function of n, as shown in Table. 1, strongly improved the performance of algorithm with the cubic and annealing methods giving P_{soln} over 0.83 for annealing method and 0.92 for cubic method. Although, the linear variation method gives P_{soln} near to 1 for all n, it requires j of $O(n^3)$ which results in increasing search cost C compared with other methods.

In the results, the quadric variation method revealed to give average search cost of $O(n)$ giving substantial improvement over the other methods as well as the

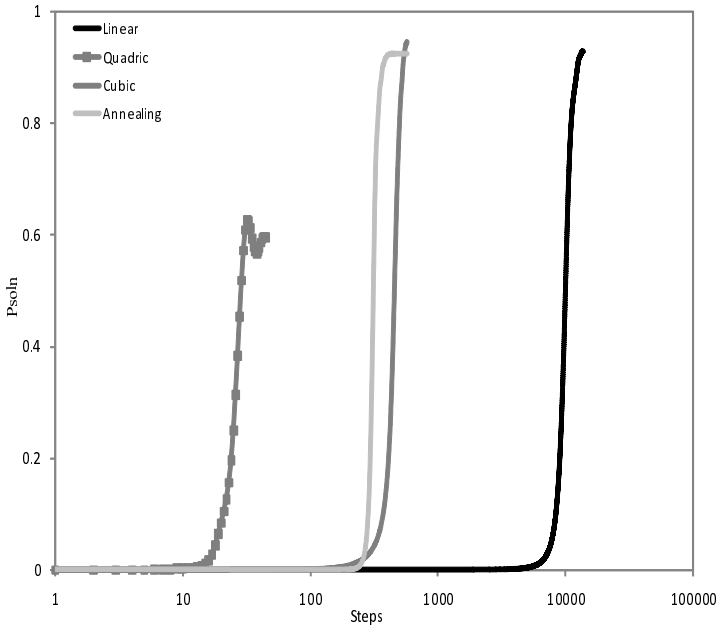


Figure 6.17: P_{soln} *vs* number of steps h for the 4 methods using improved values of parameter Δ. The figure shows the average behavior calculated over 100 random instances of 3-SAT problems with $n = 24$.

known classical methods for the problem size feasible to be simulated on the classical computers.

The experiments also have shown that using improved Δ, i.e., dependents on n gives fairly high P_{soln} values, showing better use of quantum coherence in the discrete formulation of adiabatic algorithm.

6.8.2 Resulting Search cost

In this article the search cost is defined to be the expected number of steps required to find a solution $C = j/P_{soln}$ which is a commonly used proxy for the computational cost of the discrete methods [68], pending further study of the clock rates of the used operators.

Fig. 6.8.2 and Fig. 6.8.2 compare the average search cost C for all the methods using original set and improved set of the parameter Δ, respectively. From these figures we can see that the quadric variation has search cost in both cases of Δ values below that of the other methods, where it can achieve fairly high P_{soln} between 0.5

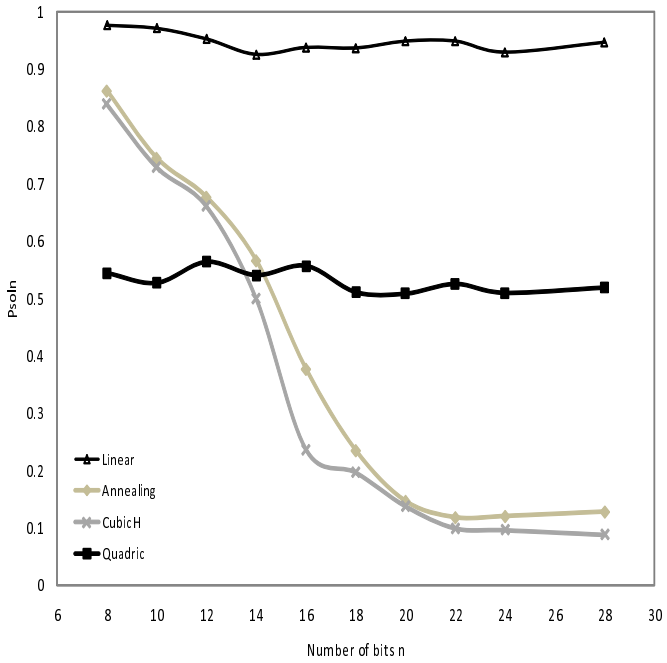


Figure 6.18: The average P_{soln} *vs.* n for the four methods using the original values of parameter Δ. The same 100 instances were solved.

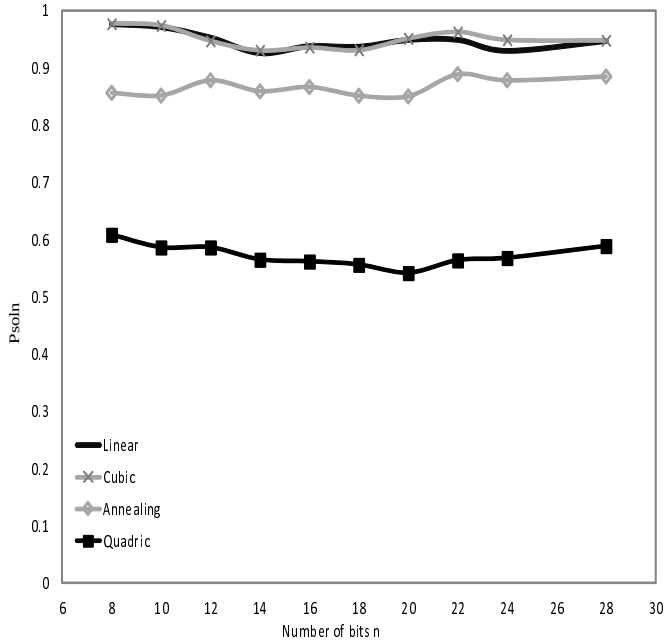


Figure 6.19: The average P_{soln} *vs.* n for the four methods using the improved values of parameter Δ. The same 100 instances were solved.

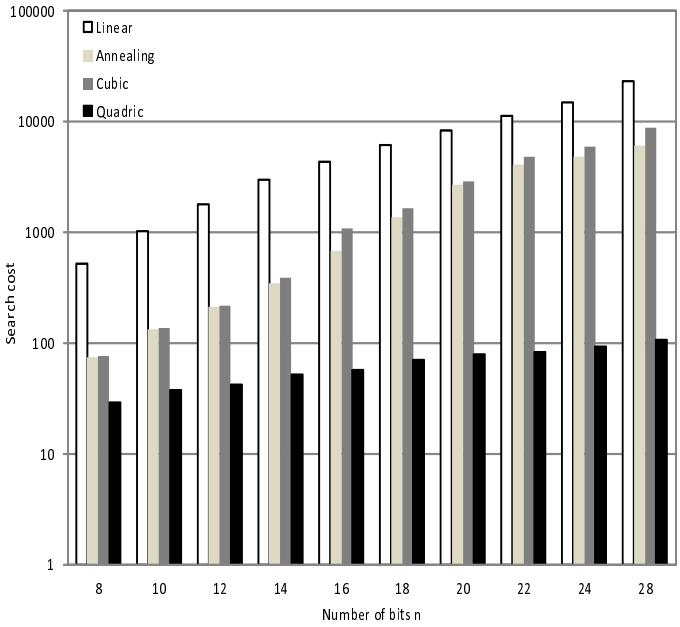


Figure 6.20: Log plot of the average search cost *vs.* n using the same instances in Figure 6.18 for the original values of parameter Δ.

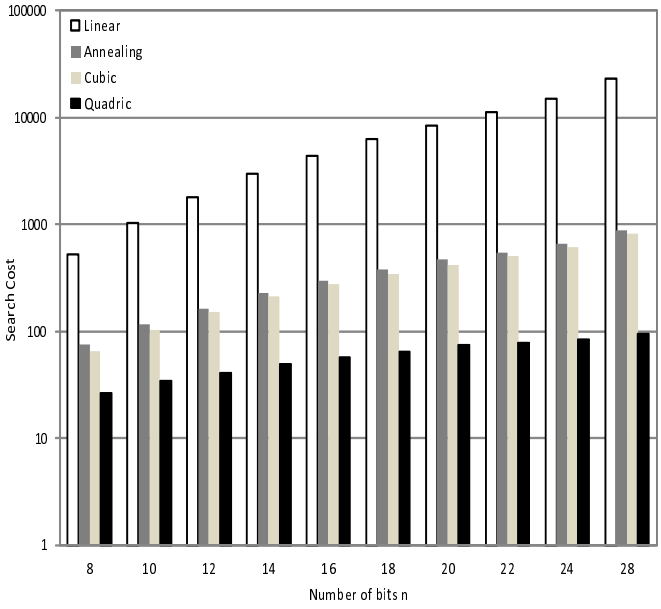


Figure 6.21: Log plot of the average search cost *vs.* n using the same instances in Figure 6.18 for the improved values of parameter Δ.

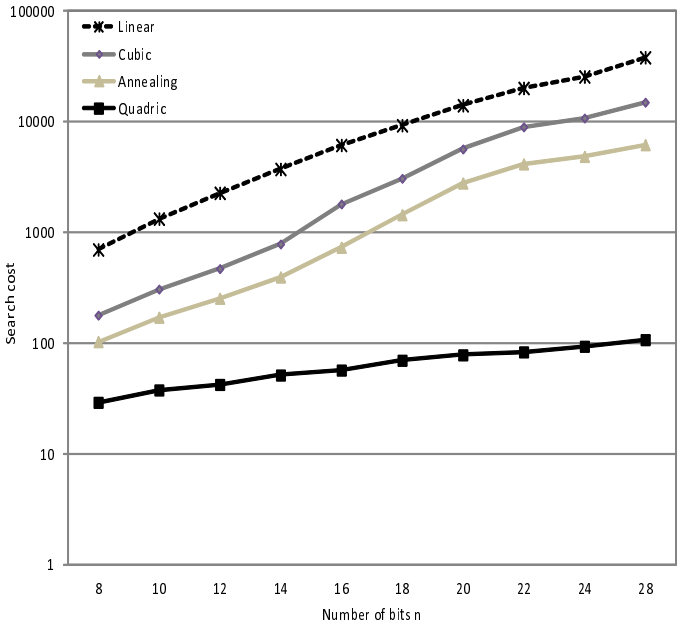


Figure 6.22: Log plot of the average search cost *vs.* n using the same instances in Figure 6.18 for the original values of parameter Δ in a line graph.

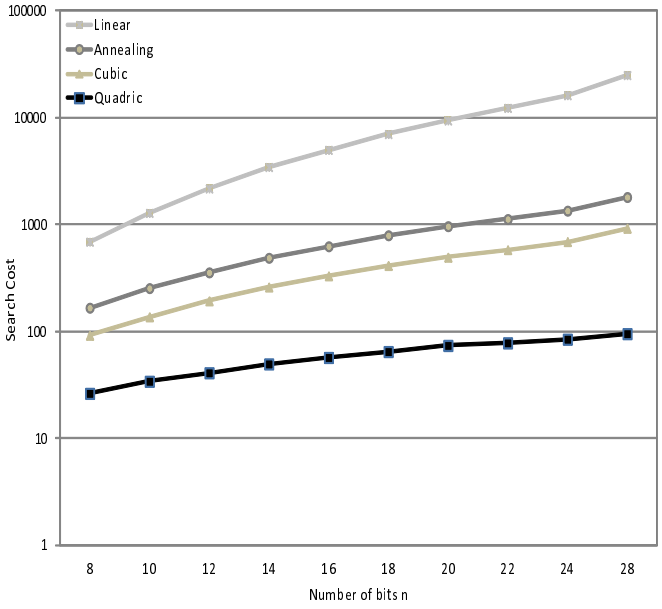


Figure 6.23: Log plot of the average search cost *vs.* n using the same instances in Figure 6.18 for the improved values of parameter Δ in a line graph.

and 0.6 with number of steps j growing only as $2 \cdot n$ which reduces the search cost to be only of $O(n)$, which is an observable improvement over the other methods if it continue for larger n beyond the limit of simulation. In contrast, cubic and annealing methods show higher search cost of $O(n^2)$ due to higher required number of steps, at least grows as n^2, in both cases of Δ values. However, the search cost reduces in fig. 6.8.2 with improved values of Δ due to the improvement of P_{soln}. Although, the algorithm with linear variation could achieve P_{soln} near 1 in most of the trials, the number of steps required j grows as n^3 results in a much higher search costs of $O(n^3)$, which is far higher than those of other methods.

For comparison at $n = 20$, Fig. 6.8.2 shows that the average search cost for the linear, cubic, annealing and quadric methods are 8424.14, 2887.41, 2722.49 and 78.64, respectively with original values of Δ, and Fig. 6.8.2 shows that the average search cost for the same methods with improved Δ are 8424.14, 419, 470, and 73.12, respectively. These comparisons reveals the improvement in the resulting search cost using quadric variation method due to reduced number of steps, also it shows the search cost reduction in the other methods cubic and annealing using the improved values of Δ due to the strong enhancement in the P_{soln} as shown in previous section.

6.9 Conclusions

Quantum adiabatic algorithm is a method of solving computational problems by evolving the ground state with a slowly varying Hamiltonian. This algorithm is a remarkable discovery because it offers new insights into the potential usefulness of quantum resources in solving combinatorial search problems. This article presents an experimental study on the discrete formulation of the adiabatic algorithm in solving 3-SAT problems. We presents a new monotonic variation method for the phase functions of the algorithm (scheduling method) named, Quadric variation method. In addition, we present an improved formula to find a better value of parameter Δ of the algorithm as linear polynomial in the problem size in order to improved the algorithm search behavior and the over all search cost. With problem sizes feasible to evaluate by using simulation on current computers, the experiments

in solving 3-SAT problems have revealed that using the quadric variation method improves on the other variation methods linear, cubic, and annealing in resulting search cost and search behavior, where it gave search cost of only $O(n)$. Also the improved values of the parameter Δ has strongly enhanced the algorithm search behavior for all the variation methods.

Summary

This thesis mainly presents a study on a novel paradigm recently presented to design a new quantum algorithm based on adiabatic theorem, this algorithm is known as quantum adiabatic algorithm. Here we manly presents a study on solving combinatorial search problems using the discrete quantum adiabatic algorithm. We aim by this research to speed up the adiabatic quantum algorithm as well as decrease the overall resulting search cost. In order to do that we proposes a new variation method for the phase functions in the adiabatic algorithm called Quadric variation method in several versions. In addition, we present a better parameter configuration for the algorithm.

Chapter 1 provides an introduction explains briefly the historical development for the ideas to find alternatives to conventional computation and how it leads to think about the quantum computation as a strongly possible alternative and the adiabatic evolution as an idea for computation. Also this chapter describe the motivation of research and specify the dissertation structure.

Chapter 2 gives an overview of the basic concepts and ideas related to this research namely the basic principles for the quantum computation such as the qubits, interference, and parallelism. At the same time, it defines the targeted combinatorial search problem, Satisfiability problem. this chapter also describes the conceptual differences between the adiabatic evolution model we study and the original quantum computation model known as circuit model.

Chapter 3 describes the novel idea behind the adiabatic quantum evolution, and how the adiabatic theorem could be used to evolve the computer system from the initial state to the target final state. This chapter also presents a numerical example

describes how to apply the adiabatic evolution method. Then it proceeds to describe the conceptual differences between the local and the global adiabatic evolution, the definition and the importance of the minimum energy gaps, and how to design a practical adiabatic quantum algorithm.

Chapter 4 reports about the results of the experiments carried out to solve instances of knapsack problems with quantum heuristic search algorithm (QHS). This chapter describes in details the operators and the parameters used in the QHS algorithm. Then it reports on the results obtained from solving random instances for three different types of the knapsack problems, finally it presents a comparison between the search behavior of the QHS and the Genetic algorithm using two types of the constrain handling methods namely Penalty Function and Random Repair. Ends with findings and conclusions

Chapter 5 describes a new variation method for the phase functions of the adiabatic quantum algorithm named Quadric variation method. This method was the first proposed version as partially monotonic variation method. A number of previously presented variation methods are briefly referenced showing the parameter configuration for each and the corresponding minimum energy gaps. Then it proceeds to outline the main functions available in the simulator we uses for the experiments. it reports on results obtained from the experimental study set up to compare the proposed method with most recently proposed variation methods such as cubic variation and linear variation. The experiments used random instances of 3-SAT problems with difference number of variables up to 20 bits. The chapter concludes with summary of findings and suggestions for future works where the experiments revealed that the quadric variation method improve on the other methods such as linear and cubic in the resulting search cost and shows stable search behavior, However more modification and better settings for the parameter are expected.

Chapter 6 extends the presented work and describes the second version of the quadric variation method as a complete monotonic variation method. In addition, it describes the improved formulas we suggests to find a values for the parameter Δ of the algorithm in order to improve the algorithm search behavior and decrease the resulting search cost. Then it proceeds to present a discussion for the experimental

results of solving 3-SAT problems up to 28 bits, and presents a comparison for the resulting probability of finding solution, corresponding minimum energy gaps, and the search costs for all the methods. This chapter ends by concluding the findings and the results reflecting the experiences gathered along the research process, where the results have revealed that using the monotonic quadric variation method improves on the other variation methods linear, cubic, and annealing in resulting search cost and search behavior, where it gave search cost of only $O(n)$. Also the improved values of the parameter Δ has strongly enhanced the algorithm search behavior for all tested variation methods.

Bibliography

[1] D.P. DiVincenzo, Quantum computation, Science 270 (1995) 255:261.

[2] G. Moore. Cramming more components onto integrated circuits. Electronics, 38(8):114-117, 1965.

[3] E. Rieffel and W. Polak. An introduction to quantum computing for nun-physicists. ACM Computing Surveys, 32(3):300-335, 2000.

[4] J. Bell. On the problem of hidden variables in quantum mechanics. Reviews of Modern Physics, 38(3):447, 1996.

[5] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer, Proceedings of the Royal Society of London A 400, pp. 97-117 (1985 (1985) 97-117.

[6] R. Feynman. Feynman Lectures on Computation. Addison-Wesley, Reading, 1996.

[7] D. Deutsch, and R. Josza. Rapid solution of the problems by quantum computation. In proceeding of royal sociality of London , vol 449, pp. 553-558,1992.

[8] T. Hogg, D Portnov , Quantum Optimization. Information Science, Vol 128, P 181-197, (2000).

[9] T. Kato. On the adiabatic theorem of quantum mechanics. J. Phys. Soc. Jap., 5:435:439, 1951.

[10] A. Messiah. Quantum Mechanics. John Wiley and Sons, New York, 1958.

[11] M. A. Nielsen and I. L. Chuang. Quantum computation and quantum information. Cambridge University Press, Cambridge, 2000.

[12] G. Rose. The first real Quantum CPU, D-Wave Systems, Inc. 2008.

[13] Nakayama, S. (2008, April). Lectures on Advanced Quantum Computation, Advanced lectures conducted from Kagoshima University, Japan.

[14] S. MORITA, Faster Annealing Schedules for Quantum Annealing, Journal of the Physical Society of Japan vol. 76, No. 10, October, 2007.

[15] P. Dirac. The principles of Quantum Mechanics. Clarendon Press, Oxford, 1947.

[16] J. Cirac and P. Zoller, Quantum computations with cold trapped ions. Physical Review letters, vol: 74, 4091-4094, 1995.

[17] I. Enej, Adiabatic Quantum Computation, Lecture in University of Ljubljana, Faculty for Mathematics and Physics, May 2010.

[18] D. Bacon, Decoherence, Control, and Symmetry in Quantum Computers, arXiv:quant-ph/0305025 (2001) .

[19] A. Younes, Dr. Thesis, Faculty of Science, University of Birmingham, England (2001).

[20] P. DiVincenzo. Quantum computation. Science, 270:255?261, 1995.

[21] P. Shor. Polynomial-Time Algorithms for Prime Factorization on a Quantum Computer, SIAM J. Computing, vol. 26, 1997.

[22] D. Deutsch, and R. Jozsa, Rapid solutions of problems by quantum computation, Proc. R. Soc. Lond. A 439 (1992) 553.

[23] L. Grover, Fast Quantum Mechanical Algorithm for Database Search, ACM Symp. Theory of Computing, ACM Press, New York, 1996, pp. 212-219.

[24] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum Computation by Adiabatic Evolution, quant-ph/0001106.

[25] E. Farhi, A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. Science, 292:472-476, 2001.

[26] E. Farhi, J. Goldstone, S. Gutmann A Numerical Study of the Performance of a Quantum Adiabatic Evolution Algorithm for Satisfiability, quant-ph report no. 0007071 (2000).

[27] A.M. Childs, E. Farhi, J. Goldstone, S. Gutmann, Finding cliques by quantum adiabatic evolution, Quantum Information and Computation 2, 181 (2002).

[28] V. N. Smelyanskiy, and D. A. Timucin. Simulations of the adiabatic quantum optimization for the set partitioning problem. Los Alamos preprint quant-ph/0112143, 2002.

[29] E. Farhi and S. Gutmann, An Analog Analogue of a Digital Quantum Computation, quant-ph/9612026; Phys.Rev. A 57, 2403 (1998).

[30] J. Roland, and N.J. Cerf. Quantum search by local adiabatic evolution. Physical Review A, 65:042308, 2002. Los Alamos preprint quant-ph/0107015.

[31] S. Das, R. Kobes, and G. Kunstatter, Adiabatic Quantum Computation and Deutsch's Algorithm, Phys.Rev. A65 (2002).

[32] W. van Dam, M. Mosca, and U. Vazirani, Simple Proof of Equivalence between Adiabatic Quantum Computation and the Circuit Model, in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 01), pp. 279:287, Los Alamitos, CA, 2001.

[33] E. Farhi, J. Goldstone, and S. Gutmann, Quantum Adiabatic Evolution Algorithms with Different Paths. quant-ph /0208135(2002).

[34] M. A. Nielsen and I. L. Chuang. Quantum computation and quantum information. Cambridge University Press, Cambridge, 2000.

[35] B. Reichardt. The quantum adiabatic optimization algorithm and local minima. In Proc. 36th STOC, pages 502:510, 2004.

[36] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev. Adiabatic quantum computation is equivalent to standard quantum computation. In Proc. 45th Annual IEEE Symp. on Foundations of Computer Science (FOCS), pages 42:51, 2004.

[37] J. Avron, R. Seiler, and L. G. Yaffe. Adiabatic theorems and applications to the quantum Hall effect. Commun. Math. Phys., 110(3), 1987.

[38] E. Farhi, J. Goldstone, and S. Gutmann. Quantum adiabatic evolution algorithms versus simulated annealing. Technical report, lanl-arXive quant-ph/0201031, 2002.

[39] G. Rose, The first real Quantum computer, D-Wave Systems, Inc. 2008.

[40] M. Stephen on March 30, 2009, Hardy paradox, The Quantum Interference, <http://granades.com/2009/03/30/hardys-paradox-or-the-economist-is-dismal-at-science >.

[41] C. Brunker, and A. Zeilinger, Youngs experiments and the fitness of information, Philos. Trans. R. Soc. London, Vol 360, 1061:1069, 2002.

[42] P. Feyman and Hibbs, Quantum Mechanics and Path Integrals Mc Graw Hill, New York, 1965.

[43] C. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, Experimental Quantum cryptography. Journal of Cryptography, 5(1):3-28,1992.

[44] J. Bell . On the porblem of the hidden variables in quantum mechanics . Reviews of Modern Physics, 38(3):447, 1966.

[45] H. Azuma. Building partially entangled states with Grovers amplitudes amplification process. International Journal of Modern Physics C,11(3): 469-484,2000.

[46] C. Bennett, G. Brassard, C. Crepeau, R. Josa, A. Peres, and W. Wooters. Telporting an unknown quantum state via dual classical and Einstein-Podoslky-Rosen chanels. Physical Review Letters,70:1895-1899, 1993.

[47] M. A. Nielsen, and I. L. Chuang. Quantum computation and quantum information, Cambridge University. Press (2000).

[48] R. Raussendorf, H.J. Briegel. A One-Way Quantum Computer, Phys. Rev. Lett. 86 (2001).

[49] R. Raussendorf, J Harrington, and K. Goyal. A fault-tolerant one-way quantum computer, Annals of Physics 321 (2006).

[50] M.K. Vandersypen, M. Steffen, G Breyta, C.S. Yannoni, M.H Sherwood, and I.L Chuang. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance, Nature 414 (2001).

[51] Lu C.Y., Browne D.E., Yang T., Pan J.W.: Demonstration of Shor's quantum factoring algorithm using photonic qubits, Phys. Rev. letter. 99 (2007).

[52] B.P. Lanyon, T.J. Weinhold , N.K. Langford, M. Barbieri, D.F.V. James, A. Gilchrist, and A.G. White, Experimental demonstration of Shor's algorithm with quantum entanglement, Phys. Rev. Lett. 99 (2007).

[53] N. Shigeru, Study on Adiabatic Quantum Computation, Kyushu, Japan, Aug-2009 (presentation).

[54] A. Mitra, A. Ghosh, R. Das, A. Patel, and A. Kumar, Experimental implementation of local adiabatic evolution algorithms by an NMR quantum information processor, arXiv:quant-ph/0503060v2 (2005).

[55] A. Tulsi, A. Mitra, and A. Kumar. Experimental NMR implementation of a robust quantum search algorithm, arXiv:0912.4071v1 (2009).

[56] D. P. DiVincenzo, Quantum computation, Science 270 (1995) 255–261.

[57] E. Rieffel, and W. Polak. An introduction to quantum computing for nun-physicists, ACM Computing Surveys, 32(3):300-335, 2000.

[58] T. Hogg, Highly structured searches with quantum computers, Phys. Rev. Let. 80 (1998) 2473–2476.

[59] T. Hogg, D. Portnov, Quantum Optimization, information Science 128 181-197, (2000).

[60] T. Hogg, Quantum search heuristics, Phys. Rev. A 61 (052311) (2000).

[61] I. Rechenberg, Optimization of Technical Systems according to the Principles of Biological Evolution, Frommann-Holzboog, Stuttgart (1973).

[62] A. Eiben, A.E. Smith, Introduction to Evolutionary Computing, Springer. J.E. (2003).

[63] T. Hogg, Single-step quantum search using problem structure, 1998. Los Alamos preprint quant-ph/9812049 .

[64] M.R. Gary, and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.

[65] D. Pisinger, Where are the hard knapsack problems? Technical Report 2003/08, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark. 2003.

[66] S. Martello, and P. toth, Knapsack problems-Algorithms and computer Implementations, Hon. Wiley and Sons Ltd, 1990.

[67] T. Imabeppu, S Nakyama, and S. Ono. Experimental study on Pair Swap Strategy in Quantum-Inspired Evolutionary Algorithm, Journal of Advanced computational Intelligence and Informatics, vol.4 No.4 2006.

[68] T. Hogg, Adiabatic Quantum Computing for Random Satisfiability Problems, quant-ph/0206059, 2004.

[69] M. El-Fiky, S. Ono, and S. Nakayama, . iscrete Adiabatic Quantum Computation with Quadric Variation, in proceeding of The Second World Congress on Nature and Biologically Inspired Computing (NaBIC2010), pp. 2550-2555, IEEE-Explore, Kita-kushu, Japan, 2010,

[70] I.L. Chuang, M. Steffen, W. van Dam, T. Hogg, and G. Breyta, Experimental implementation of an adiabatic quantum optimization algorithm, Phys. Rev. Lett. 90 (2003).

[71] R. Harris, A.J. Berkley, and J. Johansson. Implementation of a Quantum Annealing Algorithm Using a Superconducting Circuit, quant-ph/020601, 2007.

[72] B. Altschuler, H. Krovi, and J. Roland. Anderson localization casts clouds over adiabatic quantum optimization, arXiv:0912.0746v1 (2009).

[73] K. P. Marzlin and B.C. Sanders. Inconsistency in the application of the adiabatic theorem. Physical Review Letters, 93:160408, 2004.

[74] B. Reichardt. The quantum adiabatic optimization algorithm and local minima. In Proc. 36th STOC, pp. 502:510, 2004.

[75] D. M. Tong, K. Singh, L.C. Kwek, and C. Oh, Quantitative conditions do not guarantee the validity of the adiabatic approximation. Physical Review Letters, 95:110407, 2005.

[76] A. M. Childs, E. Farhi, and J. Preskill, Robustness of Adiabatic Quantum Computation, Phys. Rev. A 65 012322 (2002).

[77] A.J. Leggett et al., Dynamics of the Dissipative Two-State System, Rev. Mod. Phys. 59, 1-85 (1987).

[78] J.P. Paz and W.H. Zurek, Quantum Limit of Decoherence: Environment Induced Super-selection of Energy Eigenstates, Phys. Rev. Lett. 82(26), 5181-5185 (1999).

[79] A. Ambains, and O. Regev. An Elementary Proof of the Quantum Adiabatic Theorem, 2006. arXiv:quant-ph/0411152v2.

[80] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness.W. H. Freeman, San Francisco, 1979.

[81] S. Kirkpatrick and B. Selman, Critical behavior in the satisfiability of random boolean expressions. Science, 264:1297-1301, 1994.

[82] J. Roland and N. J. Cerf. Quantum search by local adiabatic evolution. Physical Review A, 65:042308, 2002. Los Alamos preprint quant-ph/0107015.

[83] M. El fiky, S. Ono, S. Nakayama, Study on Quantum Heuristic Search in an NP-Hard Problem, in proceeding of ICROS-SICE International Joint Conference 2009, pp.2550-2556, International Congress Center, Fukuoka, Japan: 2009.

[84] W.van, Dam M. Mosca, and U. Vazirani, How Powerful is Adiabatic Quantum Computation?, quant-ph/0206003 v1 (2002).

[85] S. Das , R. Kobes, and G Kunstatter Energy and Efficiency of Adiabatic Quantum Search Algorithms, quant-ph/0204044v4; Phys.Rev. A 57, (2003).

[86] M. Steffen, W. van Dam, T. Hogg, G. Bryeta, I. Chuang, Experimental implementation of an adiabatic quantum optimization algorithm, Phys. Rev. Lett 90 (2003) 067903.

[87] E. Farhi and S. Gutmann, An Analog Analogue of a Digital Quantum Computation, quant-ph/9612026; Phys.Rev. A 57, 2403 (1998).

[88] A. Mackworth, Constraint satisfaction. In S. Shapiro, editor, Encyclopedia of Artificial Intelligence, pages 285-293. Wiley, NY, (1992).

[89] B. Altschuler, H. Krovi, J. Roland, Anderson localization cast clouds over adiabatic quantum optimization, arXiv:quant-ph/0912.0746v1 (2009).

[90] E. Farhi, J. Goldstone, D. Gosset, S. Gutmann, B. H. Meyer, and P. Shor, Quantum Adiabatic Algorithms Small Gaps and Different Paths, arXiv:0909.4766v1 [quant-ph] (2009).

[91] U. Vazirani. On the Power of Quantum computation, Philosophical Transactions of the Royal Society of London A, Volume 356, Number 1743, pp. 1759-1768 (1998).

[92] M. El-fiky, S. Ono, and N. Shigeru, Study on Discrete Adiabatic Quantum Computation in 3-SAT problems, in The sixteenth International Symposium on Artificial life and Robotics, Oita, Japan, 2011, pp.49-652.

[93] M. Boyer, G. Brassard, P. Hoyer, and A. Tapp. Tight bounds on quantum searching. In T. Toffoli et al, editors, Proc. of the Workshop on Physics and Computation (PhysComp96), pp. 36-43, Cambridge, MA, 1996. New England Complex Systems Institute.

[94] G. Brassard, P. Hoyer, and A Tapp. Quantum counting. In K. Larsen, editor, Proc. of 25th Intl. Colloquium on Automata, Languages, and Programming (ICALP98), pages 820-831, Berlin, 1998. Springer.Los Alamos preprint quant-ph/9805082.

[95] S. M. Maurer, Tad Hogg, and B. A. Huberman. Portfolios of quantum algorithms. Physical Review Letters, 87:257901, 2001. Los Alamos preprint quant-ph/0105071.

[96] A.M. Childs, E. Farhi, and J. Preskill, Robustness of Adiabatic Quantum Computation, Phys. Rev. A 65 012322 (2002).

[97] A.J. Leggett, Dynamics of the Dissipative Two-State System, Rev. Mod. Phys. 59, 1-85 (1987).

[98] J.P. Paz and W.H. Zurek, Quantum Limit of Decoherence: Environment Induced Super-selection of Energy Eigenstates, Phys. Rev. Lett. 82(26), pp. 5181-5185 (1999).

[99] M. El-fiky, S. Ono, and N. Shigeru, Study on Speeding Up of Adiabatic Quantum Computation in Satisfiability Problems, = in 11th Asian Quantum Information Science Conference AQIS11, Busan, Koria, Vol. 16, pp. 121-123. 2011.

[100] M. El fiky, S. Ono, and N. Shigeru, Study on Discrete Adiabatic Quantum Computation in 3-SAT problems, International Journal of Artificial life and Robotics (ISAROB), Vol. 16, pp. 106-111, 2011.