OPTIMAL ANNEALING PATHS FOR ADIABATIC QUANTUM COMPUTATION

by

Navid Yousefabadi

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science

at

Dalhousie University
Halifax, Nova Scotia
December 2011

DALHOUSIE UNIVERSITY

DEPARTMENT OF PHYSICS AND ATMOSPHERIC SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "OPTIMAL ANNEALING PATHS FOR ADIABATIC QUANTUM COMPUTATION" by Navid Yousefabadi in partial fulfillment of the requirements for the degree of Master of Science.

Dated: December 9, 2011

Supervisor: _____

Readers: _____

_____

DALHOUSIE UNIVERSITY

DATE: December 9, 2011

AUTHOR: Navid Yousefabadi

TITLE: OPTIMAL ANNEALING PATHS FOR ADIABATIC QUANTUM COMPUTATION

DEPARTMENT OR SCHOOL: Department of Physics and Atmospheric Science

DEGREE: M.Sc. CONVOCATION: May YEAR: 2012

Signature of Author

Table of Contents

List of Tables

List of Figures

Abstract

Shor's algorithm shows that circuit-model quantum computers can factorize integers in polynomial time – exponentially more efficiently than classical computers. There is currently no analogous algorithm for Adiabatic Quantum Computers(AQCs). We illustrate through a number of factorization problems that a naive AQC implementation fails to reveal an exponential speed up. An exponential speed up does become evident with the optimization of the AQC evolution path utilizing existing optimisation approaches. We reduce the computation time even further by optimization over heuristically-derived parametrised functions. Finally, we improve our own results by exploring two-dimensional paths, and give arguments that using more dimensions in the search space can enhance the computational power to an even greater extent.

List of Abbreviations and Symbols Used

AQCs	Adiabatic Quantum Computers
AQC	Adiabatic Quantum Computation
QND	Quantum non-demolition
$\mathcal{T}$	The annealing time or the time that the system is evolved
δ	Minimum energy gap between ground state and the first excited state
Δ	Instantaneous energy gap between ground state and the first excited state of evolving Hamiltonian
$\mathcal{F}$	Fidelity (the probability of finding the answer)
QAB	Quantum Adiabatic Brachistochrone
EL	Euler-Lagrange

Acknowledgements

I would like to thank a patient teacher, and a good friend, my supervisor Professor Kyriakidis. Without his helps none of this would have been possible. His supports always gave me strength to work. He was always available to answer questions, and I'm so grateful that I had the opportunity to gain from his knowledge. I also appreciate the assistance I've received from the other members of the Kyriakidis group, specifically Eduardo Vaz and Catherine J. Stevenson. I would also like to thank my parents, who made sacrifices so that I can be successful.

Halifax, December 2011

Navid Yousefabadi

Chapter 1

Introduction

The field of Quantum Computation has become very popular in recent years. Since the formulation of Shor's algorithm in 1994 [1], scientists have been enthusiastic to make a Quantum Computer. In Sec. 2.1.1, we describe the circuit model of Quantum Computation. Shor's algorithm, which has an exponential speedup over classical algorithms, is based on this model of computation. In Sec. 2.1.2, we list the essential criteria of a feasible Quantum Computer, and through examples we show that experimentalists have not yet been able to realize these criteria in a physical system.

In Sec. 2.2.1, the adiabatic theorem is described, where the system stays in the same eigenstate as it is time-evolved adiabatically. In Sec. 2.2.2, Adiabatic Quantum Computation (AQC) is described, where a system with an easily prepared ground state is adiabatically driven to a system with a complex ground state. Using the adiabatic theorem, if the evolution begins in the ground state of the initial system, then it should end in the ground state of the final system, which contains the answer to a problem that is encoded in it.

In Sec. 2.2.3, the circuit model quantum computer is compared to the adiabatic quantum computer. It is described that these two models are equivalent in the sense that both are universal quantum computers; the logic of any computer algorithm can be simulated on their circuits. On the other hand, a circuit model quantum computer is experimentally difficult to realize, whereas an adiabatic quantum computer has an inherent noise resistance, which makes it an experimentally realizable model. A circuit model quantum computer is known to be exponentially more efficient than a classical computer for some problems. However, the power of an adiabatic quantum computer is not determined. These issues are described in more details in Sec. 2.2.4.

In Chapter 3, the Ising spin glass model is described, which is a system of spins with long range random interactions in an external field. Finding the ground state of this complex system is computationally intractable. We use this model for AQC,

where the final Hamiltonian is one of a spin glass, and the initial Hamiltonian is constructed so that its ground state can be easily prepared. Although the Ising spin glass is known not to be universal, a very large class of problems, like factorization of integer numbers, can be solved with it. In Sec. 3.2, we write the multiplication circuit for integer numbers in binary representation, and derive some constraints that relate the input bits to the output bits. We show that a penalty function can be written in terms of the bits, such that its minimum satisfies the constraints. This penalty function is then converted into an Ising spin glass Hamiltonian. In this way, the factorization problem is encoded in the spin glass, where the ground state (all constraints satisfied) represents the answer to the problem.

In this work, we solve some instances of integer factorization using AQC to find the scaling of the computation time with the problem size. Note that we use the terms "computation time", and "annealing time" interchangeably. These terms denote the time required to time-evolve the system from the ground state of the initial Hamiltonian to some final state of the final (problem) Hamiltonian. We will consider various interpolation (annealing) paths. The annealing time will be largely governed by the target fidelity, usually taken to be 0.9. That is, we will anneal the system slowly enough to obtain a 0.9 probability for the final state to be found in the ground state of the final Hamiltonian. We show that, although naive AQC fails to reveal an exponential speedup over the classical case, there are optimal annealing paths that can exponentially improve the computation time. We solve some instances of the factorization problem with simulations of the Ising spin glass. We time-evolve the Hamiltonian numerically to find the exact final state, and compute the probability of measuring the system to be in the ground state of the final Hamiltonian. These allow us to make judgements about the annealing time, and the annealing path.

A one-parameter evolution scheme with a constant speed (the naive AQC) results in an exponential scaling of annealing time with problem size (Sec. 4.1.1). In Sec. 4.1.2, by using the Quantum Adiabatic Brachistochrone method, we tune the speed in the one parameter evolution with the gap, and show that the scaling of the annealing time with problem size will reduce to a polynomial. (The annealing time is inversely proportional to the energy gap between the ground and first-excited states.)

In Sec. 4.1.3, we reduce the evolution times even further by using an optimal interpolation in a one-parameter evolution scheme, with a heuristically-derived parametrised function. In Sec. 4.2.1, we use a two-parameter evolution scheme, and explore the two-dimensional gap, where paths with larger energy gaps are derived, while the speed is tuned with the gap value. In Sec. 4.2.2, we derive two improved parametrised functions that can find larger gaps. In each step (going from one-parameter evolution to two parameter evolution, and changing the parametrised function to a more suitable one) the computation time is reduced.

Finally, we conclude with a summary and some comments future work and the generalisation of our work to problems beyond factoring.

Chapter 2

Adiabatic Quantum Computation

2.1 History

In the beginning of twentieth century, a revolution happened in physics. This was after new experimental techniques were developed, and scientists could reach the microscopic domain. It turned out that classical physics could not explain microscopic phenomena. By 1925, Heisenberg and Schrödinger could successfully unite the experimental findings in to the theory of quantum mechanics. [2]

Not happy with the ideas that were introduced with quantum theory, Einstein, Podolsky and Rosen [3] used an ingenious thought-experiment (EPR paradox) to show how unsatisfying the interpretations of quantum mechanics were. Einstein believed that, based on physical reality, the measurement outcome of an attribute of a particle should be independent of the outcome of measurements on another particle. After this argument, known as Einstein's locality principle, some alternatives to quantum mechanics theory were proposed. J. S. Bell [4] predicted a testable inequality showing the disagreement between quantum theory and its alternatives [5]. All these events led to the perception of quantum entanglement. This phenomenon can be explained in this example: Consider a particle at rest and with spin 0 decaying into two spin-$\frac{1}{2}$ particles. These particles should move in opposite directions to conserve linear momentum, and to conserve angular momentum the two particle system should have zero angular momentum. Two experimentalists, A and B, decide to measure the z component of angular momentum of the two particles separately. If A measures first and reports a spin up, then B should report a spin down. Quantum entanglement says, no matter how far apart these two particles are, the measurement of A has instantaneous effect on the measurement outcome of B. [6]

At the same time, some developments also happened in computer science. In 1965, Moore predicted that every two years the number of transistors that can be placed on an integrated circuit doubles, it is now known as Moore's law. Amazingly,

his prediction held true since the 1960s. Scientists realized that this trend will end in the first decades of the twenty-first century, because, as electronic devices become smaller and smaller, quantum phenomena become more effective in the functionality of devices. After this, a theory of quantum computation was developed, where instead of classical physics, quantum mechanics is used to perform computation. [7]

In 1994, Peter Shor wrote a quantum algorithm [1]that factorizes integer numbers in polynomial time, where on classical computers the best algorithms for integer factorization run in super-polynomial time [8]. This exponential speed up of quantum computers over classical ones made them so promising and pursuable. The power of quantum computer is said to be a direct result of entanglement between quantum states [9].

2.1.1 Circuit Model Quantum Computation

In classical computers, the basic unit of information used for computation is the bit. In quantum computers, an analogous concept is used, where it is called a qubit. A bit is a state that can be 0 or 1; the corresponding states for quantum computers (qubit) are $|0\rangle$ and $|1\rangle$. The difference between a bit and a qubit is that a qubit can be in a linear superposition of states, e.g.,

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Where α and β are complex numbers. In classical computation, one can measure a bit to see if it is a 0 or 1. When measuring qubits one can get 0 with a probability of $|\alpha|^2$ and 1 with a probability of $|\beta|^2$. We therefore require that $|\alpha|^2 + |\beta|^2 = 1$. This strange property of qubits that they can be in a continuum of states until being observed, plays an important role in the power of quantum computers. As does entanglement, whereby a two-qubit state can exist in a non-classical, non-separable state such as $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$.

The NAND gate is a universal gate for classical computers; it can be used to compute any function. A universal set also exists for quantum computation, where an arbitrary function can be computed with a quantum circuit using these gates. This set consists of the Hadamard, Phase, CNOT, and $\pi/8$ gates. Note that this universal set is not unique. The Hadamard, Phase, and $\pi/8$ are single qubit gates, this

means they act on one qubit. The matrix form of these gates can be written as follows:

$$Hadamard \quad \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$Phase \quad \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

$$\frac{\pi}{8} \quad \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

As an example, if in a quantum circuit the Hadamard gate (H) is applied to a qubit state $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the result will be:

$$H|\Psi\rangle = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle$$

The new qubit state has a different probability of being in state $|0\rangle$ and $|1\rangle$.

In computation theory, controlled gates are in this form: 'If A is true, then do B', where they act on two qubits. CNOT is a controlled gate, a very useful quantum gate with two input qubits; a control qubit A, and a target qubit B. The action of CNOT gate is as follows: If the control qubit is set to $|1\rangle$ then the target qubit is flipped, and if the control qubit is set to $|0\rangle$ then the target qubit is not flipped. The matrix representation of this gate is:

$$CNOT \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

If the CNOT gate is applied to a qubit in state $|\Phi\rangle = \alpha|01\rangle + \beta|10\rangle$ then the new state is:

$$CNOT|\Phi\rangle = \alpha|01\rangle + \beta|11\rangle$$

In quantum computation, these universal gates are combined in a quantum circuit to approximate arbitrary functions to arbitrary accuracy. Sophisticated problems need more complex quantum circuits, and because of the non-intuitiveness of the quantum states, making an efficient quantum algorithm is a really hard task.

Utilizing these quantum gates, two efficient fundamental classes of algorithms have been discovered so far. The first class, which has an exponential speed up over the best known classical algorithms, is based on Shor's quantum Fourier transform. It provides algorithms for factoring and discrete logarithm problems. The second class, which has a quadratic speed up over the best possible classical algorithms, is based on Grover's search algorithm. This algorithm is important because so many classical algorithms are based on search techniques, therefore fast quantum analogs can be made with Grover's algorithm. For a more in-depth description, on circuit model quantum computation, see [10]

2.1.2 Physical realization

In order to build a quantum computer, at least three criteria must be satisfied.

The first one is scalability; we must be able to have as many qubits as needed in the system without exponentially increasing the resources. One of the main obstacles in making practical quantum computers is that it is difficult to use so many qubits and at the same time have a controllable system. As the number of qubits increases the perseverance of coherence becomes more and more difficult in the quantum system.

The second criteria is universal logic; we must be able to perform an arbitrary computation with a quantum computer. We should be able to perform arbitrary transforms which could be done by the set of universal gates. For this we should be able to implement the set of logical gates on the suggested hardware fault tolerantly. An alternative way of doing the computation which doesn't need the gates and is as powerful as the circuit model is adiabatic Quantum computation, which is the primary focus of this dissertation.

The third criteria is correctability; we need to be able to perform quantum error correction in which we initialize the states and then measure them to detect the effect of the environment. So correctability means the ability of preparing an initial state and the ability of measuring the state. Sometimes measuring the state is equivalent

to preparation of initial state. These kind of measurements are called Quantum non demolition measurements [11]. By QND measurement, the state doesn't change after performing the measurement and therefore one needs to prepare fewer initial states.

A universal quantum computer must have all of these three conditions together, and while performing the computation, the state should be coherent. This is somewhat challenging due to technological limitations, because all systems exhibit some amount of decoherence [12]. Therefore, simultaneously initiating, computing, and measuring an scalable system in an isolated situation from the environment, where there is decoherence, is an experimentally difficult job. Technological improvements are making this job more realizable each day.

Some physical systems have been investigated to become the future quantum computer. Scientists are using any advancement in technology to improve their models, and make a practical quantum computer. Photonic quantum computation is one of those models [13], where photons are used as qubits. A photon's state does not interact strongly with the medium so it does not face decoherence. Photons can be produced by lasers and detected by photomultipliers. Single qubit gates can be implemented by using beamsplitters and phase shifters. For creating entanglement between two photons in order to perform a CNOT gate, nonlinear media were first used. Atom-photon interactions in optical cavities was used later to further reduce absorption losses due to optical nonlinearities. A breakthrough was made, when it was found that the computation could be done by one photon source and detector without ever needing a nonlinear media [14]. Afterwards the focus directed toward finding efficient photon sources and detectors for the purpose of making a scalable quantum computer.

Another model is trapped ion quantum computers [15], where ionic states are used as qubits. Their states conserve a good coherence, which is needed for computation. Ionic qubits can be initialized by optical pumping, and can be measured by florescence. The ions are trapped via electric fields, and entangled states are created by ionic interaction. Qubits can be coupled via the collective ionic motion induced by laser pulses. In this way, the qubits are manipulated for computation. If this system is scaled to a larger number of qubits, handling the collective motion of ions becomes difficult. One method to improve on this problem is to separate the ions

from each other and couple them by photonic interactions [16]. This model can be made in the same manner with atoms too. In this case, the atoms are trapped by counter-propagating laser beams, creating an optical lattice. By using laser beams, the geometry can be precisely defined [17].

Another model is nuclear magnetic resonance quantum computers [18], where the spin of the nucleus is used as the qubit. The spin state of nucleus has a long coherence time and therefore could be a good candidate for a qubit. It can be initialized by freezing the spin state of nucleus with a strong magnetic field. By applying magnetic field pulses, the spins can be controlled, and as a result single qubit gates can be implemented. Entanglement is provided by outer shell atomic interactions, therefore allowing two qubit operations. The measurements can be made by the currents, induced from the magnetic moments.

Quantum dots are used as qubits as well [19]. There are electrostatic and self assembled dots that can be used as a qubit. In the case of electrostatic dots, there can be an array of dots defined by controlled voltages, in which each of them has an electron inside. The spin of the electron defines the qubit. The sate of the qubit can be manipulated by changing the electrostatic voltage and therefore a computation can be performed. The spin can be measured by the ability of an electron to tunnel into the dot [20]. In this model, the nuclear spin of the semiconductor produces a magnetic field which causes the coherence time of the system to decrease. It is better to use semiconductors which don't have nuclear spin, namely silicon and germanium. There has been so much development in introducing new ways to create and manipulate dots in efficient ways.

In each proposed model, by increasing the number of qubits there should be enough coherence time that the computation can be done fault tolerantly. For a particular problem, the important factors are, a) the kind of quantum error correction that is needed, b) the number of states that are needed to be initialized and measured for that error correction, c) the number of gates that are needed for a particular computation. Therefore, the required coherence time that is needed for a particular computation can be estimated. The coherence time available for each introduced hardware puts a limit on the problem size. Each day, with the progress of technology, new improvements are being achieved by working on each model [21].

2.2 Adiabatic Quantum Computation (AQC)

2.2.1 Adiabatic Theorem

This is a very important theorem of quantum mechanics. It states that if a system is initially in one of its instantaneous eigenstates and if its Hamiltonian evolves slowly enough with time and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum during the evolution, it will be found at a later time in the same eigenstate of the new Hamiltonian. In this phenomena, the Hamiltonian changes slowly with time, so time dependent perturbation theory can be used to calculate the transition probability of the system going from one eigenstate to another.

The Hamiltonian for time dependent perturbation theory can be written as $\hat{H}(t) = \hat{H}_0 + \hat{V}(t)$, where $\hat{H}_0$ is the initial Hamiltonian, and $\hat{V}(t)$ is changing slowly from 0 to t, and it changes very little in the interval $0 \leq t' \leq t$.

$$\hat{H}(t)|\Psi_n(t)\rangle = E_n(t)|\Psi_n(t)\rangle$$

Where $E_n(t)$ are the instantaneous eigenvalues and the $\Psi_n(t)$ are the instantaneous eigenstates. The solutions of the time independent Hamiltonian $\hat{H}_0$ are known.

$$\hat{H}_0|\Psi_n\rangle = E_n|\Psi_n\rangle$$

The Schrodinger equation for the time evolution of the system can be written as

$$i\hbar\frac{d|\Psi(t)\rangle}{dt} = \left(\hat{H}_0 + \hat{V}(t)\right)|\Psi(t)\rangle$$

The effect of $\hat{V}(t)$ on the system is to make it eventually undergo a transition from one eigenstate to another, by either absorption or emission of energy. According to the adiabatic theorem if there is a big enough energy gap between the instantaneous eigenstate of the system and the other eigenstates, the probability of this transition should be low. We can write the transition probability of this system from the time dependent perturbation theory to calculate this probability.

$$P_{if}(t) = \left| -\frac{i}{\hbar}\int_0^t \langle\Psi_f|\hat{V}(t')|\Psi_i\rangle\, e^{\frac{i\left(E_f - E_i\right)t'}{\hbar}}\ \mathrm{d}t' \right|^2$$

P_{if} is the first order transition probability of going from initial unperturbed state $|\Psi_i\rangle$ to another unperturbed state $|\Psi_f\rangle$, E_i and E_f are the corresponding eigenvalues. After an integration by part the transition probability becomes

$$\left(\frac{1}{E_f - E_i}\right)^2 \left| \int_0^t e^{\frac{i\left(E_f - E_i\right)t'}{\hbar}} \left(\frac{\partial}{\partial t'}\langle\Psi_f|\hat{V}(t')|\Psi_i\rangle\right) \, \mathrm{d}t' \right|^2$$

In accordance to adiabatic theorem $\hat{V}(t)$ should be small and it changes very little in the time interval so that the term $\partial\langle\Psi_f|\hat{V}(t')|\Psi_i\rangle/\partial t'$ can be considered constant, and it can be taken outside of the integral

$$P_{if} \simeq \frac{4\hbar^2}{\left(E_f - E_i\right)^4} \left|\frac{\partial}{\partial t}\langle\Psi_f|\hat{V}(t)|\Psi_i\rangle\right|^2 \; sin^2\left(\frac{\left(E_f - E_i\right)t}{2\hbar}\right)$$

The transition probability will be small if the changes in the Hamiltonian is small compared to the energy difference $E_f - E_i$. We see that when the perturbation is adiabatic no transition occurs. It should be mentioned that two approximations were used here, the perturbation and the adiabatic. When the perturbation is not weak, the system could still change adiabatically. This adiabatic approximation was made with a Hamiltonian that was split in to two parts. There are more general approximations that can be found in literature [22] [23]. If we assume $H(s)$ is the general Hamiltonian, where $s = \frac{t}{\mathcal{T}}$, and $\mathcal{T}$ is the evolution time, then the adiabatic approximation can be written as

$$\frac{max\left|\langle\Psi_f(s)|\frac{dH}{ds}|\Psi_i(s)\rangle\right|}{g_{min}^2} \ll \mathcal{T}$$

Where g_{min} is the minimum energy gap between the eigenstates $|\Psi_i(s)\rangle$, $|\Psi_f(s)\rangle$, during time-evolution.

The theorem that was explained here is the basis of Adiabatic Quantum Computation. In the next section we are going to explain how the adiabatic theorem is used as an instrument for doing quantum computation. However, in our work we do not rely on this approximation, and that is because we solve the Schrödinger equation directly.

2.2.2 Adiabatic Quantum Computation

Circuit model quantum computers work with gates, where a number of unitary operators act on individual qubits. In AQC, the whole system is time-evolved adiabatically,

from the initial state to the final state, which contains the answer.

Solving problems using AQC requires two non-commuting Hamiltonians: A final Hamiltonian H_f, whose ground state encodes the answer to the intended problem, and an initial Hamiltonian H_i, whose ground state is already known and easily prepared. Then one should time-evolve the initial Hamiltonian to the final Hamiltonian beginning in the ground state of the initial Hamiltonian. The Quantum Adiabatic Theorem states that if the evolution is adiabatic, and if it begins in the ground state of the initial Hamiltonian, then it should end in the ground state of the final Hamiltonian, which is the answer to the problem we intended to solve. The Hamiltonian $H\left(\frac{t}{\mathcal{T}}\right)$ that is evolved can be written as

$$H\left(\tfrac{t}{\mathcal{T}}\right) = X_1\left(\tfrac{t}{\mathcal{T}}\right) H_i + X_2\left(\tfrac{t}{\mathcal{T}}\right) H_f \tag{2.1}$$

where $X_1(0) = 1$, $X_2(0) = 0$, $X_1(1) = 0$, $X_2(1) = 1$, and $H\left(\frac{t}{\mathcal{T}}\right)$ should be evolved from $t = 0$ to $t = \mathcal{T}$. Therefore $\mathcal{T}$ is the annealing time or the time that it takes to find the answer to a problem that is encoded in H_f.
We can define a parametrised time $s = \frac{t}{\mathcal{T}}$, and by substituting it in Equation 2.1, write the Schrödinger equation as follows:

$$\frac{i}{\mathcal{T}}\frac{d}{ds}|\Psi(s)\rangle = H(s)|\Psi(s)\rangle$$

In this work, we do not worry about the adiabatic approximation. Because we solve the Schrödinger equation numerically, and make sure that the final state is the ground state of H_f as a component. In this procedure, no information will be gained about the adiabaticity of evolution. Our objective is to make $\mathcal{T}$ small enough (ie, maximise the speed of the computation) while retaining an appreciable probability of remaining in the ground state (ie. minimising the error). Adiabaticity in and of itself is of little concern.

2.2.3 AQC and Circuit Model Quantum Computing

The circuit model quantum computers are universal when equipped with a universal set of logic gates; the logic of any computer algorithm can be simulated on their circuits. On the other hand, any quantum circuit can be simulated with AQCs with no more than polynomial overhead [24]. In this sense, the Adiabatic Quantum Computer is at least equivalent to a circuit model quantum computer and it can be considered a Turing machine, or a universal quantum computer.

Making a commercial quantum computer is a challenging task for the scientists. Up to this date, there is no quantum computer based on circuit model with more than a handful of qubits. The record is 14 qubits, where a controlled entanglement was achieved in an ion trap system [25]. It is not practically a quantum computer, but can be considered as a quantum register suitable for gate operation. The main difficulty in making a circuit model quantum computer is that the quantum states tend to decay into an incoherent state as a result of interaction with the environment. Scientists are constantly making advancements in the physical realization of circuit model quantum computing, and in the near future a commercial quantum computer may be made.

Unlike the gate model, making hardware that AQC can be implemented on, is experimentally more realizable. This is because, as the system evolves, the energy gap provides an inherent resistance to noise [26]. If the environment's energy is kept lower than the gap energy then there is a proportionally lower probability for the system to transit to a higher state. Therefore, the AQC construction is more realizable than the circuit model.

The D-Wave company has constructed a hardware chip based on superconducting electronics. These superconducting structures shield themselves from the environment interference, providing a suitable situation for quantum effects. It was shown that the D-Waves system anneals quantum mechanically for 8 qubits [27]. Later they made a 16-qubit version, and now they have a system that uses a 128 qubit processor. Although there is no proof available yet that their large machine works quantum mechanically, but they have solved a few problems with it.

2.2.4 Efficiency of AQC

The factorization of integer numbers is not tractable on traditional computers if they are the product of big prime numbers, for example if the integer is the product of two 300-digit primes [28]. A circuit model quantum computer if implemented can solve this problem efficiently using Shor's algorithm. If integer numbers can be factorized efficiently, then the cryptographic systems in use today can be decrypted. The quantum database search, Grover's algorithm, also provides a polynomial speedup, which will have a great impact on computational power.

The computational power of AQCs is not determined yet. It is mainly because the Hamiltonian of the AQC model is intractable, and consequently there is no rigid formulation that can relate the AQC's computation time with the problem size. However, we know that entanglement is the key ingredient to the quantum computation power [29], and AQC directly uses the entangled qubits to anneal the system. There are some arguments about the power of this machine in literature. In a paper by Vazirani [30], it is stated that AQC can be used to gain quadratic speedup over classical search algorithms, and later the Grover bound was recovered for AQC [31]. There is even hope that AQCs has the potential to solve NP-complete problems polynomially, where people solve random instances of NP-complete problems in polynomial time [32] [33]. In these works the AQC was simulated on a traditional computer, and the NP-complete problems that were solved with the simulated AQC, were limited to small sizes. Later, in several other similar works, it was shown that conventional AQC fails to provide an exponential speedup for solving NP-complete problems.

In this dissertation, we similarly simulate AQC (the spin glass Ising model AQC), with which we solve integer factorization instances (noting that instead of using the adiabatic approximation, we evolve the Hamiltonian in time and find the exact state of the system). Although we are limited in the problem size, we show that even if the conventional AQC fails to provide an exponential speedup, one can find optimal annealing paths that can exponentially improve the computation time. The Hamiltonian we use is the one implemented by D-Wave. This Hamiltonian is known to not be universal. However, a very large class of problems can be solved with Hamiltonian,

including the important problem of integer factorization, and this is the problem on which we will focus.

Chapter 3

Ising Spin Glass AQC

3.1 Ising spin glass

The Ising model was introduced by Ising in his 1924 PhD thesis [34]. His aim was to show a phase transition in this model. The one-dimensional Ising model was solved by Ising, where he could not see phase transition, and he thought this model is not capable of explaining phase transition. Later it was found that in 2 dimensions or more this model does exhibit a phase transition.

The model consists of spins that have states of either $+1$ or -1. The spins are arranged in a lattice, and each spin interacts with its nearest neighbour. The Hamiltonian of this model can be written as:

$$H = -\sum_{<i\,j>} c_{ij}\sigma_i\sigma_j - \sum_j h_j\sigma_j$$

Where $\sigma = \pm 1$ assigns the spin value to each site, c_{ij} is the interaction strength between adjacent sites $< ij >$, and h_j is an external field. The one and two dimensional Ising model is exactly solvable and it's partition functions are known [35].

The Ising spin glass is the Ising model with random long range two spin interactions. Our objective in solving the Ising spin glass is to find the spin configuration for which the enegy is minimum (the ground state). Many useful problems like factorization can be encoded in an Ising spin glass, where the ground state of the spin glass encodes the answer to the problem. In Sec 3.2, we explain the details of encoding foctorization problems into an Ising spin glass Hamiltonian.

We choose the system to be annealed in AQC, to be a spin glass, because not only can it encode useful problems, but also there is evidence that quantum annealing can be experimentally performed on this system. It was shown that, for probing the lowest energy configuration of a two-dimensional spin glass, quantum annealing is much faster than classical annealing [36].

The final Hamiltonian of Eq. 2.1 that encodes the problem is chosen to be a spin glass:

$$H_f = -\sum_{i\,j} c_{ij}\sigma_i^z\sigma_j^z - \sum_i h_i\sigma_j^z \tag{3.1}$$

Where σ^z is the Pauli matrix in z-direction, and the c_{ij} and h_i are chosen to solve a specific problem. In AQC we need two non-commuting Hamiltonians, so the initial Hamiltonian H_i should not commute with the final Hamiltonian H_f. This is because the initial state (the ground state of the initial Hamiltonian) can be written as a superposition of all the eigenstates of the final Hamiltonian. This way the initial state contains all the possible answers, and after annealing it converges to the ground state of the final Hamiltonian. The ground state of the initial Hamiltonian should also be easily prepared. We choose the initial Hamiltonian as:

$$H_i = \sum_{i=1}^{N} \sigma_i^x$$

Where N is the number of spins used in the problem, and σ^x is the Pauli matrix in x-direction. The Pauli matrices do not commute $[\sigma^z, \sigma^x] = 2i\sigma^y$, therefore H_i and H_f do not commute. The ground state of H_i is a spin configuration polarized in negative x-direction, because all σ^x terms have a positive sign. The N-body ground state of H_i is a simple tensor product of each spin in the state $|-x\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Therefore the initial state is a superposition of all possible configurations of spins, and it is easy to prepare.

3.1.1 Tractability of the Ising spin glass

The infinite two-dimensional Ising Hamiltonian on a square lattice, with all interactions equal to 1, and without an external field, was solved by Onsager in 1944 [35]. But a two-dimensional glass with more general interactions or an external field has not been solved yet. The ground state of a graph with n spins, can be found among 2^n possible spin configurations. If one should check each spin configuration individually to find the ground state, then the number of possibilities grow exponentially with problem size. Problems like this are considered intractable. Barahona [37] reduced an NP-hard problem to the problem of finding the ground state of a spin glass on a planar cubic lattice with each vertex of degree three, within an external field, and

proved that this problem is NP-hard. In the same paper, he reduced another NP-hard problem to the three-dimensional spin glass on a two-level grid with interactions restricted to -1, 0, 1, and proved its NP-hardness.

3.2 Factoring with the Ising Hamiltonian

In this section we are going to explain how the factoring problem is encoded to an Ising spin glass Hamiltonian [38]. To factor p as a product of a and b, we represent the numbers in binary, and write the Boolean circuit for the multiplication of a and b. The binary output of this multiplication is equal to the binary representation of p. By running the circuit in reverse we can get the unknown inputs from the known outputs.

We are going to derive some constraints with which the spin glass Hamiltonian of the problem can be constructed. As an example, consider the multiplication of two 4-bit integers (a_3, a_2, a_1, a_0) and (b_3, b_2, b_1, b_0):

$$
\begin{array}{cccccccc}
 & & & & a_3 & a_2 & a_1 & a_0 \\
 & & & & b_3 & b_2 & b_1 & b_0 \\
\hline
 & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & \\
 & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & \\
\hline
p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

The first bit of output is $p_0 = a_0b_0$, which imposes a constraint between inputs a_0, b_0 and the output p_0. This constraint has the logic of the $\wedge$ (and) gate and the truth table is given in Table 3.1, which lists the allowed combinations of inputs and outputs. This constraint can be written as a Boolean valued function $C_\wedge(a, b, c)$, that evaluates to true for combinations that satisfy the $\wedge$ gate, and false otherwise.

The next bit of output is $p_1 = (a_1b_0 + a_0b_1) \mod 2 \equiv (t + t') \mod 2$, where t and t' come from $C_\wedge(a_1, b_0, t)$ and $C_\wedge(a_0, b_1, t')$ respectively. We additionally need a carry bit c to be passed on to the next bit p_2, because $t + t'$ can be as large as 2. We can define t, t' as inputs and p_1, c as outputs. The logic is that of the Half Adder gate, shown in Table 3.1. It enforces the constraint $t + t' = p + 2c$, which can be denoted

a	b	p
0	0	0
0	1	0
1	0	0
1	1	1

t	t'	p	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

t	t'	t''	p	c
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 3.1: Truth tables for multiplication. The left one is the $\wedge$ gate, the middle one is the Half adder, and the right one is the Full adder

as $C_{2A}(t, t', p, c)$. Therefore, p_1 can be defined with this condition:

$$C_{2A}(t, t', p_1, c) \ \wedge \ C_{\wedge}(a_1, b_0, t) \ \wedge \ C_{\wedge}(a_0, b_1, t')$$

The condition on the next bit p_2 is $t + t' + t'' = p + 2c$, it can be denoted as $C_{3A}(t, t', t'', p, c)$, has the logic of the Full Adder shown in Table 3.1. Then p_2 and its carries are determined by $t+t'+t''+c$, where $C_{\wedge}(a_2, b_0, t) \wedge C_{\wedge}(a_1, b_1, t') \wedge C_{\wedge}(a_0, b_2, t'')$. By introducing $(r = t+t'+t'') \mod 2$, the condition on p_2 can be written with Full- and Half-adders as follows:

$$C_{3A}(t, t', t'', r, c') \ \wedge \ C_{2A}(r, c, p_2, c'')$$

These constraints enforce $t + t' + t'' + c = p_2 + 2(c' + c'')$, where c' and c'' are the carry bits that will be passed to the next bit.

We can extract constraints for the rest of the bits in the same manner, and this way the relationships between the input bits and output bits and the intermediate bits can be derived.

For each constraint $C(x)$ that was defined over a set of Boolean variables x, a penalty function $E(x)$ can be defined, such that

$$P(x) = \begin{cases} o & \text{if } C(x) \\ \geq o + 1 & \text{if } \neg C(x) \end{cases}$$

If the set of Boolean x satisfy the constraint, then the penalty function's output is o, otherwise the penalty function's output is at least $o + 1$. Therefore, $P(x)$ is minimised for all set of x that satisfy the constraints. The constraints $C_{\wedge}$, C_{2A}, C_{3A}

can be turned into penalty functions $P_\wedge$, P_{2A}, P_{3A} as follows:

$$P_\wedge(a,b,p) = \begin{pmatrix} a & b & p \end{pmatrix} \begin{pmatrix} 0 & 1 & -2 \\ 0 & 0 & -2 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} a \\ b \\ p \end{pmatrix}$$

$$P_{2A}(t,t',p,c) = \begin{pmatrix} t & t' & p & c \end{pmatrix} \begin{pmatrix} 1 & 2 & -2 & -4 \\ 0 & 1 & -2 & -4 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} t \\ t' \\ p \\ c \end{pmatrix}$$

$$P_{3A}(t,t',t'',p,c) = \begin{pmatrix} t & t' & t'' & p & c \end{pmatrix} \begin{pmatrix} 1 & 2 & 2 & -2 & -4 \\ 0 & 1 & 2 & -2 & -4 \\ 0 & 0 & 1 & -2 & -4 \\ 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} t \\ t' \\ t'' \\ p \\ c \end{pmatrix}$$

Note that the choice of penalty functions is not unique. We can test one of these functions after simplifying, and considering that the quadratic terms can be reduced to linear terms for binary variables (eg, $a^2 = a$ for $a = 0, 1$). The penalty function of $P_\wedge$ and P_{2A} can be written as:

$$P_\wedge(a,b,p) = a - 2ab - 2b + 3p$$

$$\begin{aligned} P_{2A}(t,t',p,c) =& t + 2tt' - 2tp - 4tc + t' - 2t'p \\ & - 4t'c + p + 4pc + 4c \end{aligned}$$

Next, the penalty functions can be considered as energy functions. Assuming a, b, p, t, t', c are spin qubits in the Ising glass model. The energy functions of $P_\wedge$ and P_{2A} can be written as:

$$E_\wedge(a,b,p) = \sigma_a^z - 2\sigma_a^z\sigma_b^z - 2\sigma_b^z + 3\sigma_p^z$$

$$E_{2A}(t,t',p,c) = \sigma_t^z + 2\sigma_t^z\sigma_{t'}^z - 2\sigma_t^z\sigma_p^z - 4\sigma_t^z\sigma_c^z + \sigma_{t'}^z - 2\sigma_{t'}^z\sigma_p^z$$
$$- 4\sigma_{t'}^z\sigma_c^z + \sigma_p^z + 4\sigma_p^z\sigma_c^z + 4\sigma_c^z$$

The penalty functions were chosen to have at most two-body interactions. Thus, the field h_i and coupling c_{ij} terms enforcing these two constraints in the spin glass Hamiltonian are determined. The eigenvalue of z-Pauli matrix for state $|0\rangle$ is 1, and for $|1\rangle$ is -1. Therefore the energy of $E_\wedge$ and E_{2A} is $o = 0$ for spin configurations that satisfy the $\wedge$ gate and the half adder respectively. And $o > 1$ for spin configurations that do not satisfy these conditions, which are listed in Table 3.1. In order to make a Hamiltonian that can factor two 4-bit integers, there are more constraints, where an energy function can be written for each of them. These energy functions are then added together, so that the constraints are satisfied together. The final Ising spin glass Hamiltonian that can factor the two 4-bit integers will be determined by deriving the field and coupling terms with the method that we described here.

In this work, we prepared the field and coupling terms that were required to factor 4-bit by 2-bit and 3-bit by 3-bit multiplications. In order to factor these 6-bit numbers, 17 bits are required, accounting for all the carry bits and the partial sum bits. To implement these Hamiltonians on the D-wave hardware, some intermediate connecting bits are additionally required to couple the bits that are physically far from each other on the chip. In Appendix A, the field and coupling terms of the 4-bit by 2-bit multiplication of 14×3 is shown as an example.

Chapter 4

Optimal Annealing Paths

4.1 One-Parameter Evolution Scheme

In this chapter, we are going to focus on the functional forms of X_1 and X_2 (see Eq. (2.1)), and the relationship of $\mathcal{T}$ to these functional forms. We define a path to be some curve in the X_1-X_2 plane, parameterised by time, along which the system evolves from H_i to H_f. Therefore, a linear path means $X_1 = 1 - X_2$. The Hamiltonian of a linear path can be written as

$$H(s) = (1 - X(s)) H_i + X(s) H_f \tag{4.1}$$

where $X(0) = 0$, $X(1) = 1$ and $s = \frac{t}{\mathcal{T}}$.
Now that the path is chosen to be linear, it is time to think of possible functional forms of $X(s)$. Conventionally, the function is chosen to be linear $X(s) = s$, but there are other possibilities that might be interesting to investigate. In the following sections we will look at different functional forms of X, and it will be shown that the choice of X will have a dramatic effect on the computational time $\mathcal{T}$.

4.1.1 Linear Function

It was mentioned that adiabatic theorem guarantees that the final state will be the ground state of the final Hamiltonian given sufficiently large $\mathcal{T}$ and gap Δ. It is known that as the system evolves from s = 0 to s = 1 there is a probability of being in the ground state of the final Hamiltonian. Therefore, if we want the final state to have a 100% probability of being in the ground state of the final Hamiltonian, the evolution should be completely adiabatic. As explained in Sec. 2.2.1, the Hamiltonian should change slowly enough during the evolution, and there should be a gap between the ground state energy of the Hamiltonian and the rest of the eigenvalues. It is proven that if $\mathcal{T} \to \infty$, and a path chosen such that the gap doesn't close, these conditions

are satisfied and the evolution is adiabatic and consequently the final state will be the ground state of the final Hamiltonian [39].

We should view the limit $\mathcal{T} \to \infty$ as an upper bound on the computation time, particularly if we seek only some finite probability of measuring the correct answer. As explained in Chapter 2, the Hamiltonian can be evolved by numerically solving the Schrodinger equation. And as described in Sec. 3.2, different problems such as factoring could be encoded in the final Hamiltonian. Having all the tools available let us try to solve a factoring problem. As an example, to find the factors of 1, the final Hamiltonian could be made to have the factors of 1 in its ground state. Evolving the system numerically, from the initial Hamiltonian to the final Hamiltonian with annealing time $\mathcal{T}$, the final state could be obtained. The probability P could be calculated by taking the modulus squared of the inner product of the final state $|\Psi(\mathcal{T})\rangle$ and the ground state $|\Psi_{GS}\rangle$ of the final Hamiltonian, which we call fidelity $\mathcal{F}$.

$$\mathcal{F} = |\langle \Psi(\mathcal{T}) | \Psi_{GS} \rangle|^2$$

In Fig. 4.1, the crossed points are calculated numerically and the lines are a natural cubic spline between the points. A natural cubic spline is provided by calculating the coefficients of a set of third-order polynomials, which pass through a set of points and the second derivative of each polynomial is set to zero at each point. One can see in Fig. 4.1, $P \simeq 1$ for $\mathcal{T} \geq 60$. $\mathcal{T} \to \infty$ is an upper limit to the adiabaticity condition. As also shown in the same figure, to factor 13 into 13 x 1, the annealing time needed is again finite but it is larger than the annealing time needed to factor 1.

It is a good time to think about the minimum annealing times needed to solve specific problems. In literature it is well known that according to the Landau-Zener theory [40] if $\mathcal{T} \sim \delta^{-2}$, there is a high probability of finding the ground state of final Hamiltonian at the end of evolution. δ is the minimum energy gap between the ground state and the first excited state energies during evolution.

The minimum energy gap δ plays a crucial rule in finding annealing times needed to solve problems. Landau-Zener theory suggests that if $\delta \to 0$, then $\mathcal{T} \to \infty$. That is, if there is a zero minimum gap then the evolution time should be infinitely large. It is important to look at the instantaneous energy gap Δ during evolution. At each s the Hamiltonian's matrix could be diagonalized using Lanczos methods. The

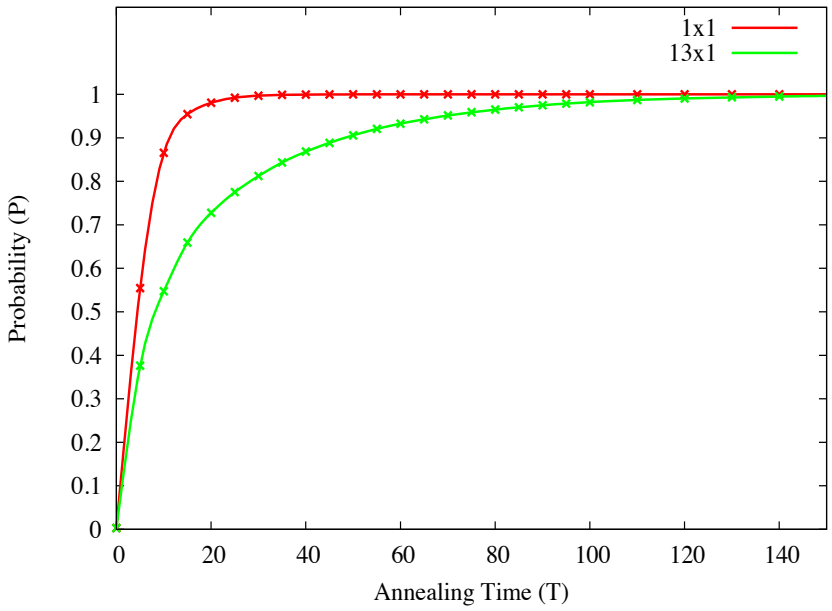

Figure 4.1: Probability of finding the ground state of the final Hamiltonian. The red line is for finding the factors of 1, And the green line is for finding the factors of 13.

instantaneous ground state and the first excited state energies could be numerically evaluated, and the difference between them will be called Δ.

In Fig. 4.2, the instantaneous energy gap of the Hamiltonian which was used to factor 1 and 13, is plotted throughout the evolution. Thus it must be clear that the gap values during the evolution play an important role in the probability values of finding the answer at certain evolution times. Evolving the Hamiltoninans numerically will give the exact final states, which allow us to have a far better description of the system than what we could get from just looking at the minimum gap. From now on the gap values and the final states will be the only things that we are going to look at. By evolving the system numerically and finding the exact final state, we don't need to be worried about the adiabatic condition. This way we avoid the approximations that theories such as Landau-Zener include.

AQC is a probabilistic computation, in a sense that the answer it finds has a probability of being the actual answer to the problem that it intends to solve. When dealing with probabilistic computers one should be careful about accuracy. However

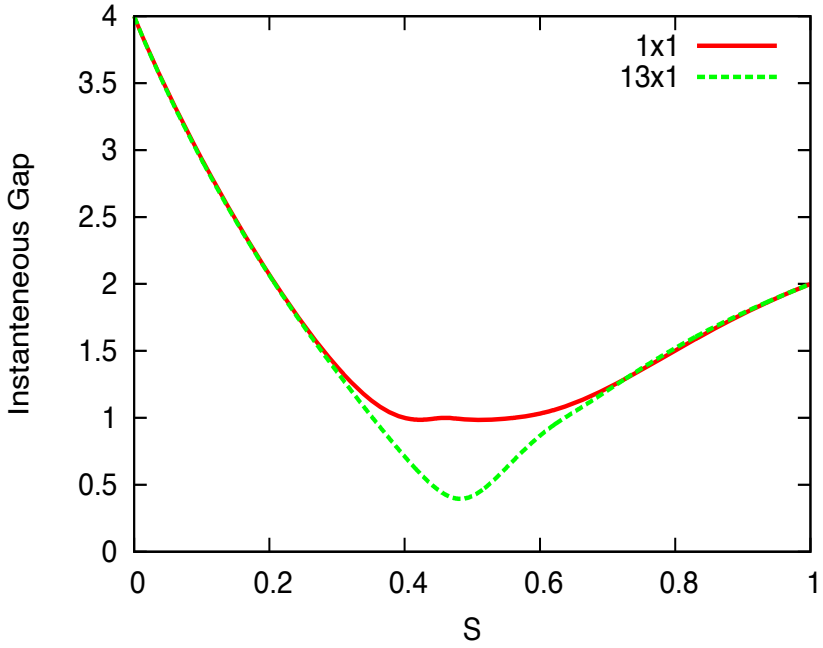

Figure 4.2: Instantaneous energy gap of Hamiltonians during evolution.

we are not going to explore this aspect of AQC. From now on we will only look at the answers that have 90% precision. Instead of P = 1 we are going to care about P = 0.9.

As described in Sec. 3.2, the factoring problems could be encoded in an Ising Hamiltonian. For this task some carry qubits are needed. It was shown that to factor a 6 bit number, 17 bits are needed in total: 6 bits represent the factors (either a 3-bit and a 3-bit number or a 4-bit and a 2-bit number) and 11-bits for the carry bits. The field and exchange terms of the 14×3 circuit (4-bit $\times$ 2-bit), is shown in Appendix A. Using our numerical simulation, we evolved the systems for different annealing times and found the annealing times for which the probability of finding the answer at the final state was 0.9. In Table 4.1, the different problems that were picked and their annealing times are listed. Note that these annealing times are obtained using a one-parameter evolution scheme. In this Table there are groups of one bit, three bit, four bit, five bit, and six bit numbers. In each group of numbers the yellow coloured box shows the annealing time of the problem that took the longest to solve. For instance in problems with size five, the hardest to factor was 21, because its annealing time is bigger than the rest of the five bit problems. In Figure 4.3 we can see the probability

Size	Circuit	Number	Annealing Time
1 Bit	3 by 3 bits	1 = 1 x 1	26.018
3 Bit	4 by 2 bits	5 = 5 x 1	24.859
	4 by 2 bits	7 = 7 x 1	19.601
4 Bit	3 by 3 bits	9 = 3 x 3	27.096
	4 by 2 bits	11 = 11 x 1	38.447
	4 by 2 bits	13 = 13 x 1	47.960
5 Bit	3 by 3 bits	16 = 4 x 4	116.383
	4 by 2 bits	20 = 10 x 2	30.931
	4 by 2 bits	21 = 7 x 3	215
	4 by 2 bits	22 = 11 x 2	15.856
	3 by 3 bits	25 = 5 x 5	168.264
	4 by 2 bits	26 = 13 x 2	16.325
	4 by 2 bits	27 = 9 x 3	34.772
	4 by 2 bits	28 = 14 x 2	11.462
6 Bit	4 by 4 bits	33 = 11 x 3	27.803
	4 by 2 bits	39 = 13 x 3	65.420
	4 by 2 bits	42 = 14 x 3	1885.5
	3 by 3 bits	49 = 7 x 7	31.728

Table 4.1: Annealing times for fidelity of 0.9 using a one-parameter evolution scheme with constant speed. The largest annealing time is highlighted in each bit sector.

plot of the hardest problems of each size. One can see that the probability increases as the annealing time increases, and for some problems this increase is faster than the others.

Shor's algorithm can factor integers in polynomial time. This was one of the greatest examples that showed quantum computers could be exponentially more efficient than classical computers. It is notable that Shor's algorithm is for circuit model quantum computers with quantum logic gates. However, for an algorithm that factorizes integer numbers on an Adiabatic quantum computer, which operates without using quantum logic gates, there is still no way to show how the computational time will scale with respect to the problem size. It is not clear whether an adiabatic quantum computer could be exponentially more efficient than a classical computer. The main issue seems to be that in the case of AQC, there is no explicit expression that can express the relationship between the annealing time needed to find the answer, and the size of the system. Having such an expression requires solving the time dependant Schrodinger equation for any large size Ising Hamiltonian. As described in Sec. 3.1.1,

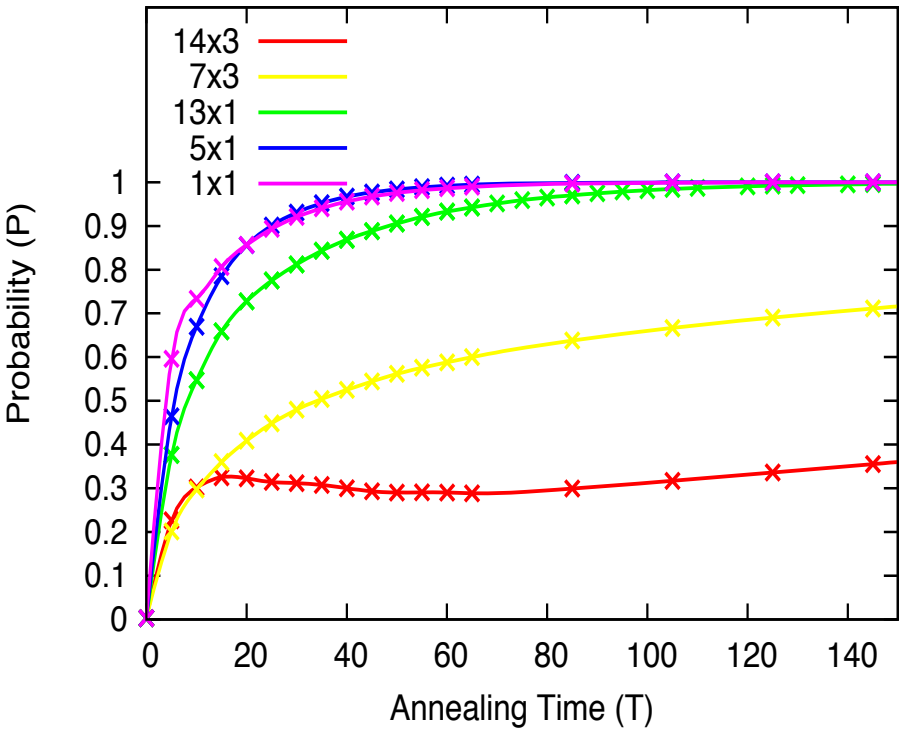

Figure 4.3: Probability of finding the ground state of the final Hamiltonians. Purple: The worst case of the problems with size 1. Blue: The worst case of the problems with size 3. Green: The worst case of the problems with size 4. Yellow: The worst case of the problems with size 5. Red: the worst case of the problems with size 6

this problem is itself intractable.

Despite the lack of analytical scaling results, it is nevertheless interesting to look at the same relationship for the small size problems that we could numerically solve. It is sensible that for our purpose which is looking at the scaling of the computational time with the problem size, we pick the hardest problem of each size.

The red points in Fig. 4.4 are the yellow colored annealing times in Table 4.1 versus their size. As mentioned above the yellow colored boxes show the annealing times needed to solve the hardest problems in each group with 90% accuracy. The red solid line in Fig. 4.4 is an exponential fit of the form $ae^{bN} + c$. Where N is the size of products (bits) and the a, b and c values are taken from Table 4.2.

It is noticeable that the computational time values that are reported here, as it was mentioned before, are just the annealing times or the times that the physical system is evolved from H_i to H_f. In the fitted plot, the annealing times that are

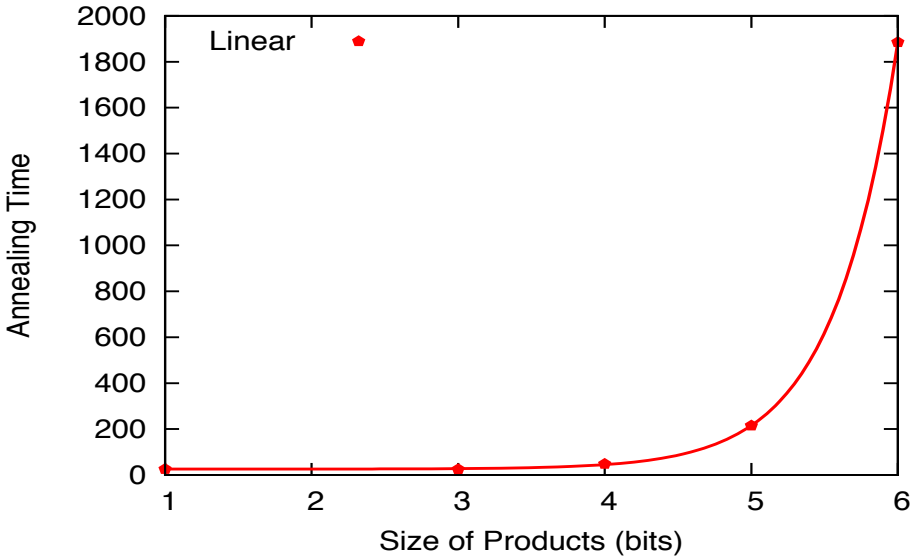

Figure 4.4: Scaling of the computation time with the problem size using a linear function of time along a linear path

reported are for solving problems with an accuracy of 90%. If there is a claim that an algorithm can factorize the integers in polynomial time, and the algorithm is a probabilistic one, then that algorithm should be able to factorize the integers with any accuracy in polynomial time. In this respect, there are some rigorous arguments about accuracy versus run time for quantum search [41].

The scaling that is talked about in the literature is for N$\rightarrow \infty$. The exponential scaling we derived was derived for problems up to N= 6. One might say, if a computational time scales polynomially or exponentially up to a certain size, it does not mean it is going to scale in the same manner afterwards. Therefore, the scaling that

rms of residuals : 2.8803
variance of residuals : 8.29614

Final set of parameters	Asymptotic standard error
$a = 0.00210702$	± 0.000229 (10.87%)
$b = 2.28179$	± 0.01792 (0.7852%)
$c = 25.6102$	± 1.802 (7.036%)

Table 4.2: Parameter values for approximating the worst annealing time points as an exponential function.

was derived here is just valid for sizes up to 6. It cannot be claimed that this algorithm scales exponentially or polynomially, because we could not look at larger sizes. By using this numerical approach nothing could be claimed about the scaling of the algorithm. Although, it is interesting to see if it is possible to reduce the exponential fit into a polynomial fit in this small range of sizes. In the following sections, we will introduce a modified algorithm that can make this happen.

4.1.2 Euler Function

In the previous Section, by using AQC simulations, a number of factoring problems in the range of sizes from 1 to 6 bits were solved. It was shown that the computation time scaled exponentially with respect to the problem size, where a one-parameter evolution scheme was used. In this section, we are going to use an optimisation method to find optimised interpolation functions. We will see that, by using optimised functions, the computation time of problems will be reduced, and will scale polynomially with respect to the sizes.

In the Quantum Adiabatic Brachistochrone (QAB) by Rezakhani et al. [42], a variational method was used to minimise the annealing time. In this method, an optimal interpolation will give rise to an optimal annealing time. From Eq. (2.1), recall the dimensionless parameter s in $X(s)$, the linear function is simply $X(s) = s$. After using the QAB method and optimizing $X(s)$, the optimal function will not be linear anymore. If we assume that along the path $\frac{\mathrm{d}\,X(s)}{\mathrm{d}\,s}$ determines the speed, then a linear function has a constant speed. QAB finds the optimal $X(s)$ by tuning the speed, such that when there is a smaller gap along the path, $X(s)$ changes more slowly with respect to s. The formulation of this theory can be derived as follows. We start from the adiabatic condition, which can be written in the following form [43]:

$$\frac{\|\frac{\mathrm{d}H(s)}{\mathrm{d}t}\|}{\Delta^2(s)} \ll \epsilon \quad \forall \quad s \in [0,1] \tag{4.2}$$

Where the norm is the Hilbert-Schmidt norm defined as $\|A\| \equiv \sqrt{Tr[A^\dagger A]}$. Taking $v(s) \equiv \frac{\mathrm{d}s}{\mathrm{d}t}$, the adiabatic condition can be written as:

$$\frac{v(s)\,\|\frac{\mathrm{d}H(s)}{\mathrm{d}s}\|}{\Delta^2(s)} \ll \epsilon$$

We know that the time needed to transverse a path is given by $\mathcal{T} = \int_0^1 \frac{\mathrm{d}s}{v(s)}$, therefore the adiabatic time functional can be defined as $\tilde{\mathcal{T}} \equiv \int_0^1 \frac{\mathrm{d}s}{v_{ad}(s)}$, where v_{ad} is the "adiabatic speed". From QAB, which is inspired from the adiabatic condition, the adiabatic speed can be defined as:

$$v_{ad}(s) \equiv \frac{\epsilon\Delta^2(s)}{\|\frac{\mathrm{d}H(s)}{\mathrm{d}s}\|}$$

Therefore the time functional can be written as follows:

$$\tilde{\mathcal{T}} = \int_0^1 \frac{\|\frac{\mathrm{d}H(s)}{\mathrm{d}s}\|}{\epsilon\Delta^2(s)}\,\mathrm{d}s \tag{4.3}$$

In order to find the optimal function(s), one needs to minimise the time. This can be done by minimising the integral of Eq. 4.3. Assuming that the integrand is a Lagrangian, one can solve the Euler-Lagrange (EL) equations to find the function(s) for which the integral is minimised. The Lagrangian can be written in this way:

$$\mathcal{L}[\frac{\mathrm{d}\mathbf{X}(s)}{\mathrm{d}s},\ \mathbf{X}(s)] = \frac{\|\sum \frac{\partial X_i}{\partial s}\frac{\partial H(\mathbf{X}(s))}{\partial X_i}\|}{\epsilon\Delta^2(\mathbf{X}(s))},$$

where the elements of $\mathbf{X}(s)$ are the control parameters that tune the interpolation. After some calculation, the EL equations for a general Hamiltonian can be written as:

$$2\frac{\partial X_j}{\partial s}\frac{\partial X_k}{\partial s}\left(\Delta\frac{\partial H_b^a}{\partial X_i}\frac{\partial^2 H_a^b}{\partial X_j\partial X_k} - 4\frac{\partial\Delta}{\partial X_j}\frac{\partial H_b^a}{\partial X_i}\frac{\partial H_a^b}{\partial X_k} + 2\frac{\partial\Delta}{\partial X_i}\frac{\partial H_b^a}{\partial X_j}\frac{\partial H_a^b}{\partial X_k}\right)$$
$$+ 2\Delta\frac{\partial^2 X_j}{\partial s^2}\frac{\partial H_b^a}{\partial X_i}\frac{\partial H_a^b}{\partial X_j} = 0,$$

where we have used the summation convention (repeated indices imply summation), and $H_b^a = \langle a|\hat{H}|b\rangle$. Solving these coupled non-linear partial differential equations is numerically hard. We should note that this optimisation is based on heuristics, and it does not guarantee an optimal path, because it does not minimise the actual annealing time.

In this section we intend to find the optimal function using a one-parameter interpolation $X(s)$. By substituting the Hamiltonian of a one-parameter interpolation, from Eq. 4.1. One can see that the Lagrangian will reduce to:

$$\mathcal{L} = \frac{\|\frac{\partial X}{\partial s}\| - H_i + H_f\|}{\epsilon\Delta^2(X)}$$

One can derive this Lagrangian, and by solving the EL equations, directly solve for X. This would be a numerical process since the gap values can only be calculated numerically. Instead of solving EL equations, which is computationally hard, one can find X from the time functional integral:

$$\tilde{\mathcal{T}} = \| - H_i + H_f \| \int_0^1 \frac{\frac{\partial X}{\partial s}}{\epsilon \Delta^2 (X)} \, \mathrm{d}s$$

Solving the EL equations for X is equivalent to finding an X for which the integral is minimised. The integral will be minimised if its integrand is made constant. This means that the numerator and the denominator should be proportional:

$$\frac{\partial X}{\partial s} = \alpha \Delta^2 (X) \tag{4.4}$$

Where α is a proportionality constant. Using QAB we can show that for a one-parameter evolution, an optimal interpolation is one whose speed along the path is proportional to the gap squared. It makes sense because as the gap gets small the adiabatic condition is hindered, and one can compensate by slowing down near those gap values. We will find the optimal functions for the factoring problems that were investigated in the previous section. To do this we must integrate the following integral to find the X values for different s values:

$$\int_0^s \mathrm{d}s = \int_0^X \frac{\mathrm{d}X}{\alpha \Delta^2 (X)}, \tag{4.5}$$

where for $s = 1$, $X = 1$ therefore the normalization constant $\alpha = \int_0^1 \frac{\mathrm{d}X}{\Delta^2(X)}$.

To calculate the integral, all we need is the $\Delta (X)$ function. Plots of $\Delta (X)$ can be seen in Fig. 4.5. In this Figure, the gap values of five different problems can be seen. It will be shown that these problems are the most difficult problem of each size after using the optimal functions to solve them. As we look at Fig. 4.5 (a), (b), (c), (d), (e) in order, we can see that the minimum gap gets smaller in these plots. These are all plotted together in Fig. 4.5 (f). In the previous section, it was explained that using a constant speed, the problems with smaller minimum gaps need a larger annealing time to be solved. This is intuitive from the adiabatic condition, because as the gap gets smaller one should change the Hamiltonian more slowly to favour the adiabaticity.

In the previous section, we used a linear interpolation with a constant speed. So for a specific problem the speed should be low enough to make up for its minimum

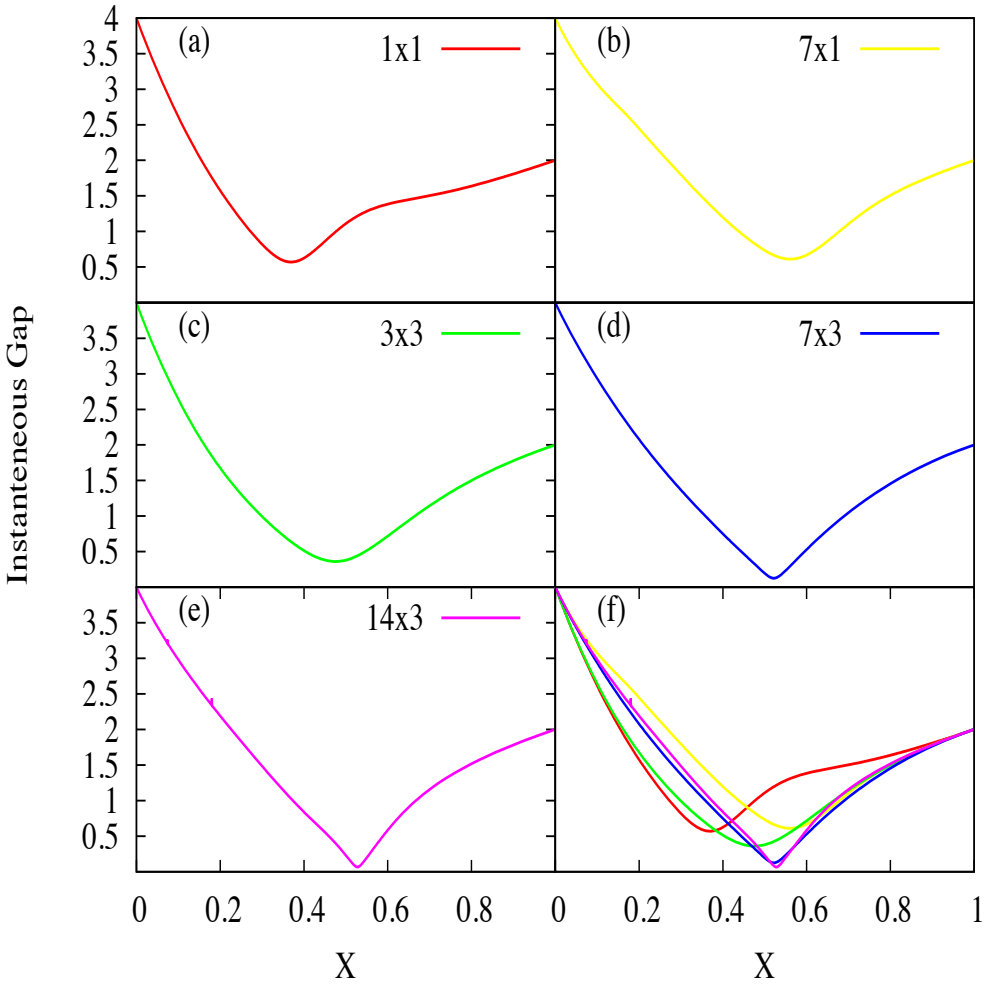

Figure 4.5: Instantaneous energy gap of Hamiltonians as a function of X values.

gap, and to provide lower speeds we needed larger annealing times. Now, by having Eq. (4.4), instead of having a constant speed we can have a dynamic one, and change the Hamiltonian faster when the gap is bigger. On the other hand in this case, it is not just the larger annealing time that favours the adiabaticity, but also $\frac{\partial X}{\partial s}$ is another factor (this factor was previously just a constant).

By using the gap values and calculating the integral of Eq. (4.5), we found the optimal functions for the problems shown in Figure 4.5. These optimal functions can be seen in Fig. 4.6. One can see that at those X values, where the gap is minimum,

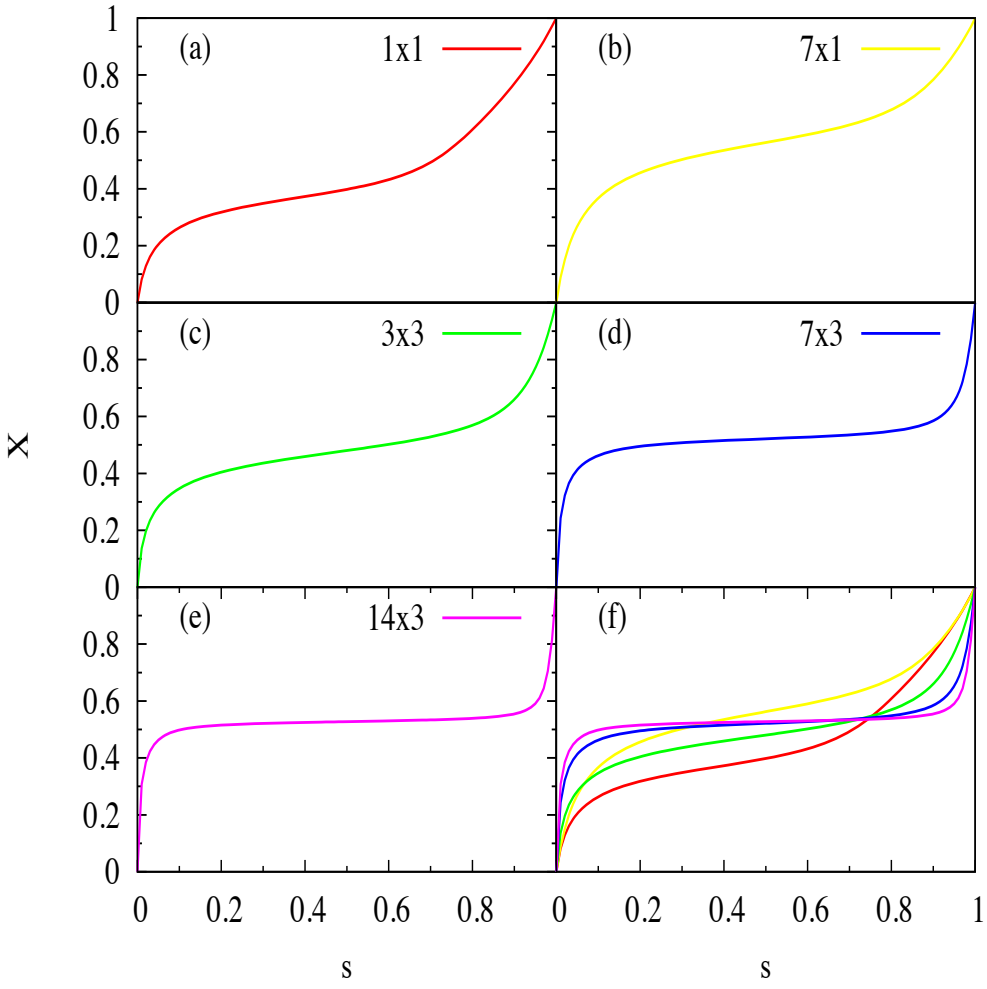

Figure 4.6: Optimal functions for a linear path, using EL equations.

the $X(s)$ slope is gradual. If one looks at Fig. 4.6 (a), (b), (c), (d), (e) in order, it can be seen that the function slope becomes more and more gradual in these plots. These are all plotted in Fig. 4.6 (f) for a better comparison.

We assume that a particular algorithm in AQC is a particular interpolation with its speed, from which the initial Hamiltonian is driven to the final Hamiltonian. Therefore the optimised interpolation represents a modified algorithm where instead of evolving the system with a constant speed, it is evolved with optimised interpolations, which are derived by EL equations.

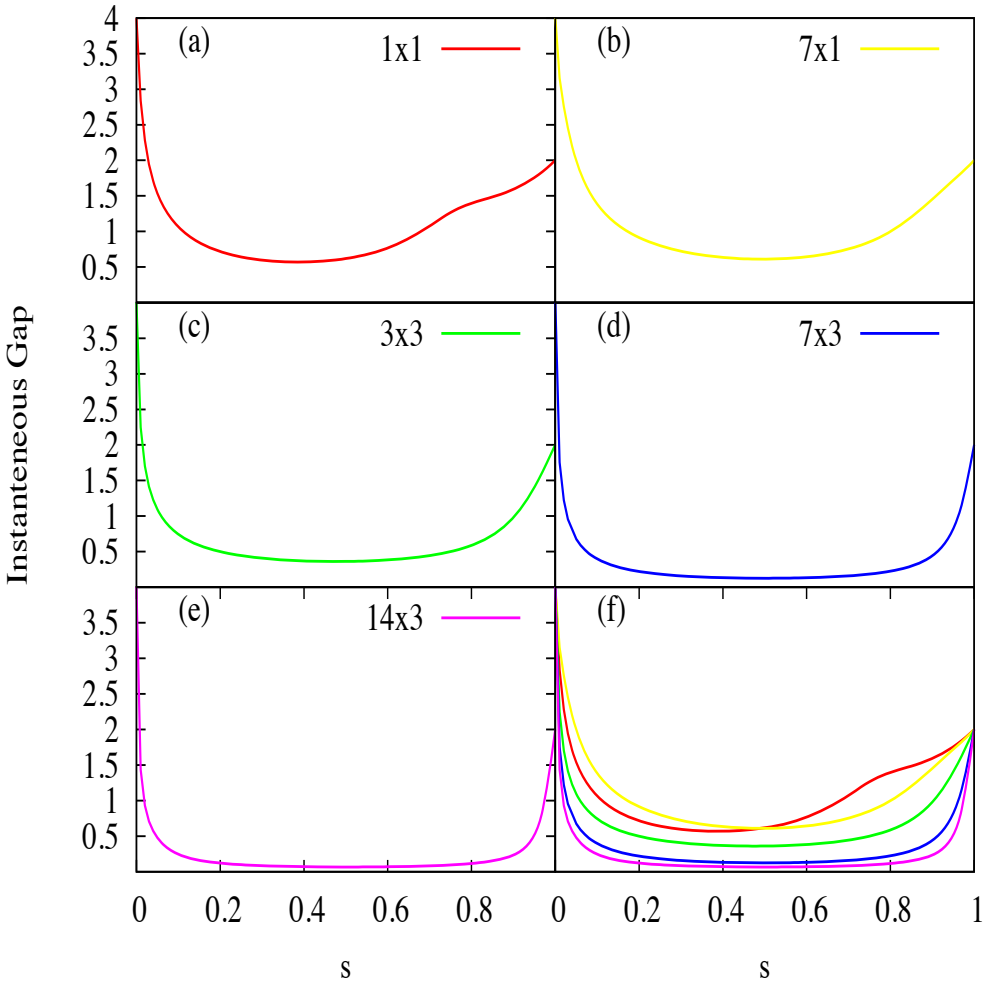

Figure 4.7: Instantaneous energy gap of Hamiltonians as a function of s.

It is interesting to look at the gap as a function of s. By looking at the plots of $X(s)$ in Fig. 4.5, one expects the gap to be small at those s values, where X has a gradual slope. In Fig. 4.7, the gaps are plotted as a function of s, for the same problems that we looked at in this section, with the same order. As expected, the X function derived from the EL equations, is designed so that whenever gap is small, it has a gradual slope. This can be easily seen, if one compares the $\Delta(s)$ plot of a problem with its $X(s)$ plot.

Size	Circuit	Number	Linear $\mathcal{T}$	EL $\mathcal{T}$
1 Bit	3 by 3 bits	1 = 1 x 1	26.018	18.4
3 Bit	4 by 2 bits	5 = 5 x 1	24.859	27.4
	4 by 2 bits	7 = 7 x 1	19.601	27.8
4 Bit	3 by 3 bits	9 = 3 x 3	27.096	44.9
	4 by 2 bits	11 = 11 x 1	38.447	32.1
	4 by 2 bits	13 = 13 x 1	47.960	35
5 Bit	3 by 3 bits	16 = 4 x 4	116.383	47.4
	4 by 2 bits	20 = 10 x 2	30.931	19.5
	4 by 2 bits	21 = 7 x 3	215	90.3
	4 by 2 bits	22 = 11 x 2	15.856	22.1
	3 by 3 bits	25 = 5 x 5	168.264	43.0
	4 by 2 bits	26 = 13 x 2	16.325	19.1
	4 by 2 bits	27 = 9 x 3	34.772	27.1
	4 by 2 bits	28 = 14 x 2	11.462	11.8
6 Bit	4 by 4 bits	33 = 11 x 3	27.803	12.1
	4 by 2 bits	39 = 13 x 3	65.420	23.4
	4 by 2 bits	42 = 14 x 3	1885.5	150.8
	3 by 3 bits	49 = 7 x 7	31.728	11.8

Table 4.3: Annealing times for fidelity of 0.9 using EL functions in the one-parameter scheme. The largest annealing time is highlighted in each bit sector.

So when $\Delta(s)$ is small, $\frac{\partial X}{\partial s}$ is also small. As it was explained, to maintain the adiabatic condition one should keep the ratio in the inequality of Eq. (4.2), as small as possible. $\Delta(s)$ is in the denominator of this ratio, therefore when Δ gets small the ratio becomes bigger. When we used the linear function ($\frac{\partial X}{\partial s}$ = const.), the only way to compensate for this was to increase the annealing time. Now, instead of increasing the annealing time, we are decreasing $\frac{\partial X}{\partial s}$, which appears in the numerator and reduces the ratio.

Using QAB, one can find optimal functions that are adjusted with the gap values throughout the evolution. This adjustment is made to keep the evolution adiabatic with the goal of a smaller annealing time. Using the QAB optimal functions, some of which are shown in Fig. 4.6, we annealed the systems to find the annealing time required to obtain a fidelity of 0.9. This was done for the same problems that were treated in Sec. 4.1.1. The results are shown in Table 4.3. The yellow colored boxes are the most difficult problem of each size. As one can see 12 out of 18 problems were improved. The QAB method performed, on average, about 33% more efficiently than

the linear function. In the same Table, if one compares the linear and EL annealing times needed for factoring $21 = 7 \times 3$ and $42 = 14 \times 3$, it can be seen that the EL annealing times are dramatically improved. The major enhancement of the EL function over the linear one is that it solves the hardest problems in a very short time.

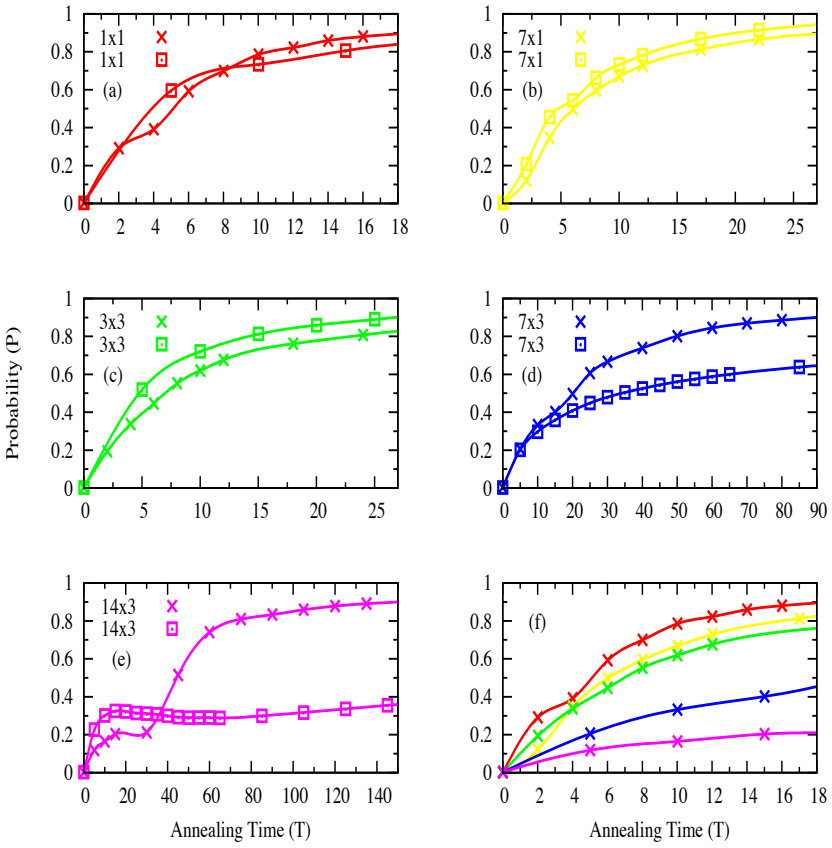

Figure 4.8: Probability of finding the ground state of the final Hamiltonian. The crossed lines show the probabilities using EL functions. The square lines show the probabilities using a linear function

In Fig. 4.8, the plots of probability as a function of annealing time are shown. These are the same problems whose gap and EL functions were shown in the previous

Figures of this section. As is evident from Table 4.3, these are the most difficult problems of each size. The crossed and the square points are the actual probabilities calculated using EL and linear functions respectively. A natural cubic spline was used to connect the points that were numerically calculated. It can be seen that for some problems such as 7×1 and 3×3, using a linear function always results in higher probabilities of finding the ground state of the final Hamiltonian. On the other hand for some problems like 14×3, if one compares the probability plots of linear and EL function, for lower annealing times the linear function gives higher probabilities, and for higher annealing times the EL function gives higher probabilities. The reason that the linear function sometimes does better than the EL functions is that the actual adiabatic condition cannot be written as an integral. The QAB is just a heuristic for the adiabatic condition and a better result is not guaranteed.

A better comparison between linear and EL functions can be made, by looking at the scaling of the computation time with the problem size, which is plotted in Fig. 4.9. This is a plot of the yellow colored annealing times in Table 4.3 versus their size. The red points and their approximation to an exponential function was shown in the previous section. The violet points, which were calculated using EL functions, were better fitted to a polynomial function of the form $aN^b + c$, where N is the size of products (bits) and a, b and c values are taken from Table 4.4. In the

rms of residuals : 3.57396
variance of residuals : 12.7732

Final set of parameters	Asymptotic standard error
$a = 0.210153$	± 0.1109 (52.77%)
$b = 3.60734$	± 0.2892 (8.017%)
$c = 17.026$	± 3.224 (18.93%)

Table 4.4: Parameter values for fitting the worst annealing time points to a polynomial function, where EL functions in the one-parameter scheme were used.

previous section the computational time scaled exponentially with problem size. In this section, the same problems were solved using our simulations, but instead of using a linear function in the evolution, we used the optimal EL functions. Utilizing this improved algorithm, the exponential scaling is reduced to a polynomial one. Although we investigated problems up to size 6, and this scaling cannot be generalized to

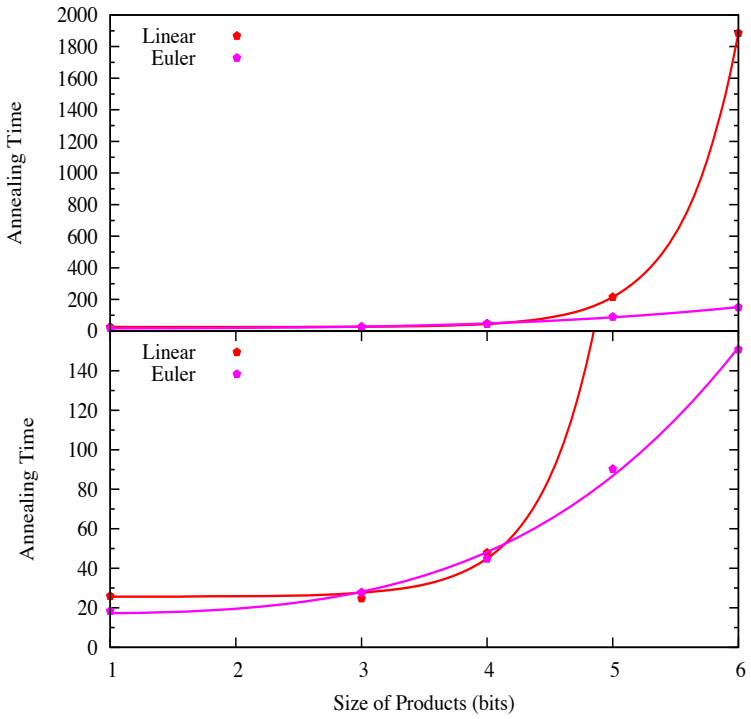

Figure 4.9: Scaling of the computation time with the problem size. Red points: using linear function of time. Violet points: using EL function. Red solid line: an exponential fit to red points. Violet solid line: a polynomial fit to violet points (Parameters given in Table 4.4).

bigger sizes, one should notice that hard problems could be solved much faster using these new functions. If AQC fails to be efficient when using linear function in a one parameter interpolation, one should consider optimal functions like EL to improve the efficiency.

One should also note that in spite of the fact that QAB can find algorithms that are much faster than the naive AQC, there is a price to pay. Finding these optimal algorithms requires knowing the gap along the evolution. Calculating the gap is intractable, it is essentially as hard as solving the original problem. The problems that we solved here had small sizes. Therefore, it was possible to find the optimal

algorithms for them. As the problem gets bigger in size, this approach will not be practical anymore. In the following sections we will use the evidence that we acquired here to develop an approach that can find the optimal algorithms heuristically, and avoid the intractability of calculating the gap. We will also find that these functions are both easier to calculate, and generally yield superior results over the EL equations.

4.1.3 Sigmoid Function

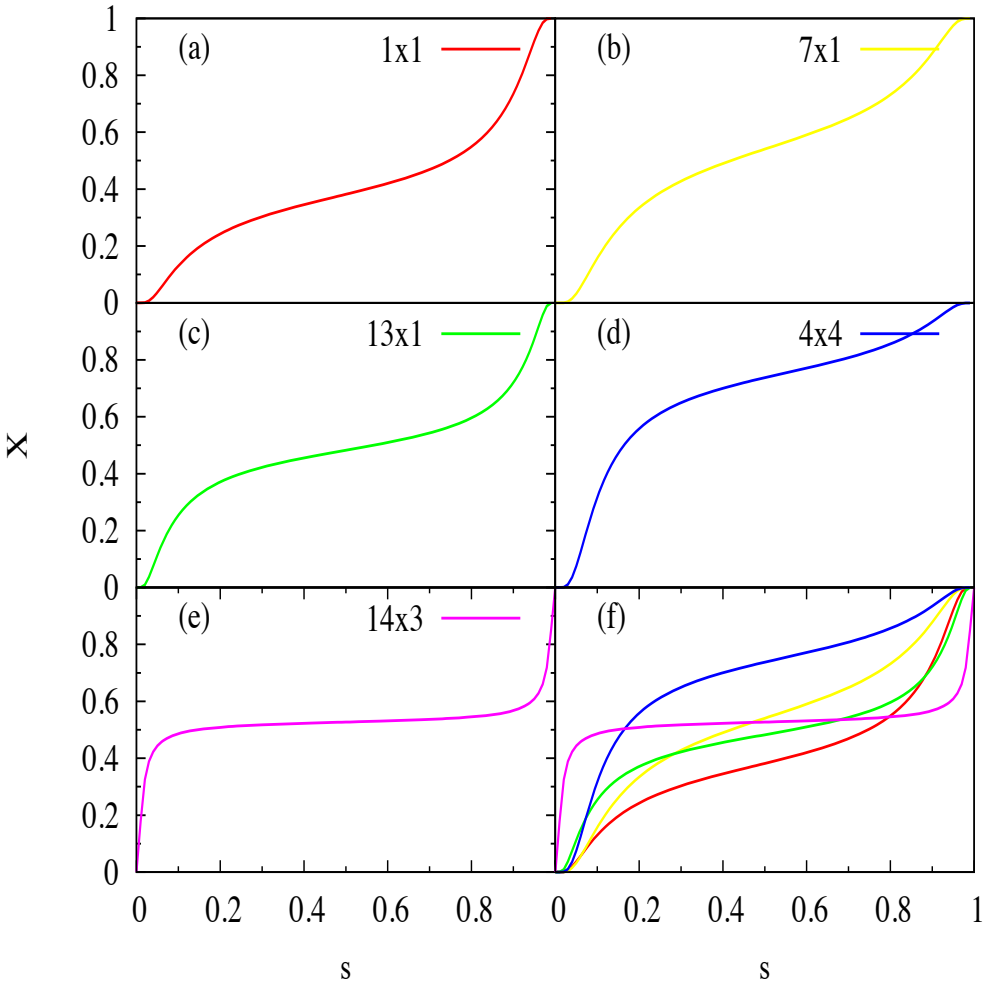

Figure 4.10: Optimal functions for a linear path, using optimised sigmoid function.

In the previous section we found the optimal functions by adjusting the slope of the function with the gap value. In this section, we are going to find the optimised functions by parameter optimisation. We introduce a heuristically-derived parametrised function and find the optimal parameters that solve the problem in the shortest time possible.

As shown in Fig. 4.6, the optimal functions that were derived with QAB method, have a generic form. The main difference between them is that the width and height of the plateau varies with the problem instance. A tunable function with the same generic form could be made. This way, one does not need to know the gap function to find the optimal annealing function. With an iterative process, the optimal parameters of the annealing function can be found.

In [44] it was shown that functions with vanishing first derivatives at the beginning and the end of evolution give increased fidelities. Adding this to the knowledge that we have about suitable functional forms, we choose the parametrized function as follows:

$$sig_{A,B}(s) = \frac{2}{\pi}\arctan\left(A\exp\left(B\tan\left(\pi s - \frac{\pi}{2}\right)\right)\right)$$

where A and B are parameters that should be determined through some optimization process. We call this function a parametrised sigmoid function. It has vanishing slopes at $s = 0$ and $s = 1$, regardless of the A and B values.

We found the optimised parameters for the same problems that were addressed in the previous sections. Using a simple steepest descents method we minimised the annealing time $\mathcal{T}$ with respect to the parameters A and B for a fixed fidelity of 0.9. The optimal functions for the hardest problem of each size are shown in Fig. 4.10. These optimal sigmoid functions are analogous to the functions that were derived using EL equations. There are slight differences between them throughout the evolution, the only big difference is that the sigmoid functions have vanishing slopes at the ends. The minimised annealing times after using optimal sigmoid functions are listed in Table 4.5. The minimum annealing time of sigmoid functions is always shorter than the annealing time of linear and EL functions, except for the case of $16 = 4\times4$. This shows the QAB does not provide us with the most optimal functions, and we can still do better. Even for the one case where the sigmoid function performed worse than the EL function, there is a good explanation. By looking at Fig. 4.11(d),

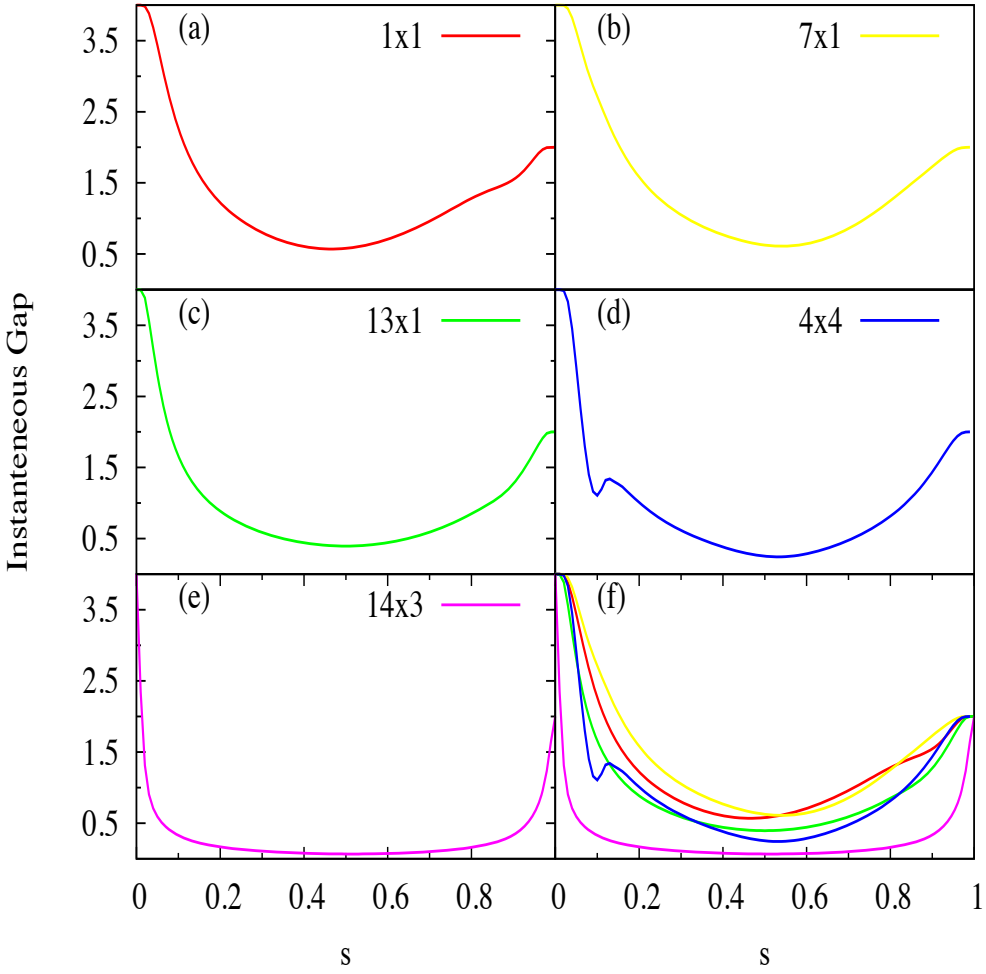

Figure 4.11: Instantaneous energy gap of Hamiltonians as a function of s, over optimal sigmoid functions.

one can see that there is an unusual bump in the gap function of 4×4. We know that the X function should be tuned with the gap. Unfortunately, our parametrised function can not adjust itself with the bump, and it is the reason that it could not improve the EL annealing time.

In order to have a more descriptive picture of the probability, we have plotted the probability as a function of annealing time in Fig. 4.12. These are the most difficult problem of each size, as highlighted in the sigmoid time column of Table 4.5. The

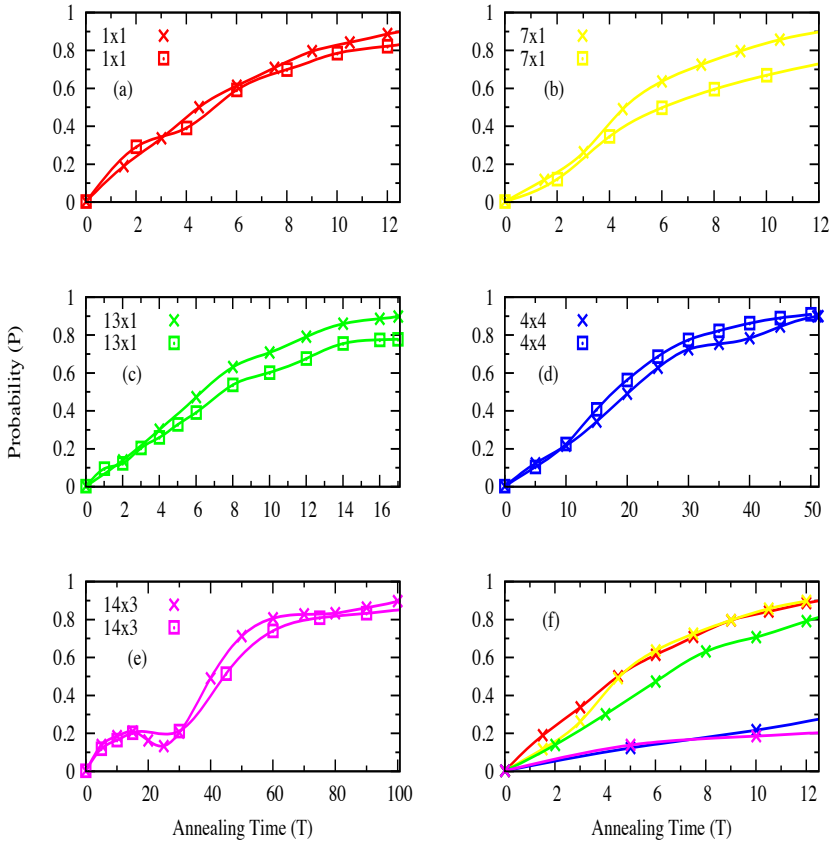

Figure 4.12: Probability density of finding the ground state of the final Hamiltonian. The crossed lines show the probabilities using sigmoid functions. The square lines show the probabilities using EL functions.

gap and optimal sigmoid function of these same problems were also plotted in the previous figures. In this plot, the points are again connected using a natural cubic spline.

To have a better comparison between the results of linear, EL, and sigmoid functions, we have plotted the largest annealing times for each size (the yellow colored boxes in Table 4.5) in Fig. 4.13. As one can see, in the worst case scenarios of each

Size	Circuit	Number	Linear $\mathcal{T}$	EL $\mathcal{T}$	Sigmoid $\mathcal{T}$
1 Bit	3 by 3 bits	1 = 1 x 1	26.018	18.4	12.435
3 Bit	4 by 2 bits	5 = 5 x 1	24.859	27.4	11.934
	4 by 2 bits	7 = 7 x 1	19.601	27.8	12.082
4 Bit	3 by 3 bits	9 = 3 x 3	27.096	44.9	12.584
	4 by 2 bits	11 = 11 x 1	38.447	32.1	15.604
	4 by 2 bits	13 = 13 x 1	47.960	35	17.131
5 Bit	3 by 3 bits	16 = 4 x 4	116.383	47.4	51.427
	4 by 2 bits	20 = 10 x 2	30.931	19.5	15.075
	4 by 2 bits	21 = 7 x 3	215	90.3	51.011
	4 by 2 bits	22 = 11 x 2	15.856	22.1	10.08
	3 by 3 bits	25 = 5 x 5	168.264	43.0	27.180
	4 by 2 bits	26 = 13 x 2	16.325	19.1	9.721
	4 by 2 bits	27 = 9 x 3	34.772	27.1	15.101
	4 by 2 bits	28 = 14 x 2	11.462	11.8	9.302
6 Bit	4 by 4 bits	33 = 11 x 3	27.803	12.1	12.022
	4 by 2 bits	39 = 13 x 3	65.420	23.4	18.319
	4 by 2 bits	42 = 14 x 3	1885.5	150.8	101.007
	3 by 3 bits	49 = 7 x 7	31.728	11.8	9.714

Table 4.5: Annealing times for fidelity of 0.9 using sigmoid functions in the one-parameter scheme. The largest annealing time is highlighted in each bit sector.

size, the sigmoid function is always faster than the two other functions. In the previous sections, it was explained that we fit the red and violet points to an exponential and a polynomial function respectively. The approximation of the blue points is done in the same manner, where a polynomial function is a better fit. After fitting the blue points to $aN^b + c$, the fitted values of a, b, and c are listed in Table 4.6.

rms of residuals : 5.05154
variance of residuals : 25.5181

Final set of parameters	Asymptotic standard error
$a = 0.0137642$	± 0.02068 (150.2%)
$b = 4.91713$	± 0.8276 (16.83%)
$c = 9.6529$	± 4.004 (41.47%)

Table 4.6: Parameter values for approximating the worst annealing time points in to a polynomial function, where sigmoid functions in a linear path were used.

It should be mentioned again that the scaling is not reliable, because we looked at the problem sizes from 1 to 6, and we can not generalize our results to bigger

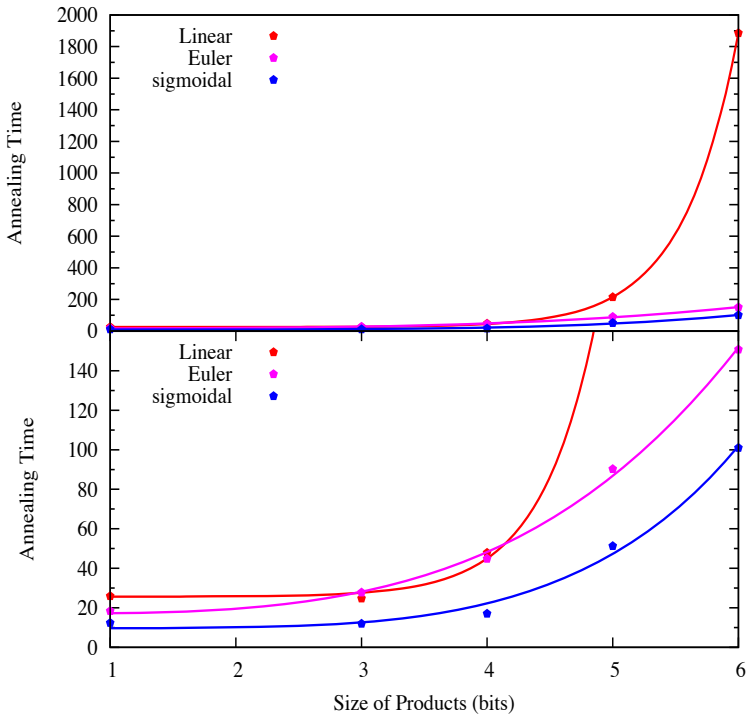

Figure 4.13: Scaling of the computation time with the problem size. Red points: using linear function in a linear path. Violet points: using EL function in a linear path. Blue points: using sigmoid function in a linear path. Red solid line: exponential fit to red points. Violet solid line: polynomial fit to violet points. Blue solid line: polynomial fit to blue points.

problem sizes. Although, if one uses optimised sigmoid functions instead of linear or EL functions, the hardest problems of each particluar size will be solved much faster. Using sigmoid parametrised functions we could improve the annealing times of the illustrated factoring problems. Further, in this method we do not need to know the gap values throughout the evolution. In the real experiment, the system could be annealed with different parameter sets of sigmoid function, the function that is well adjusted with the gap will give the answer to the problem with a higher probability, in a shorter annealing time.

4.2 Two-Parameter Evolution Scheme

4.2.1 Two Sigmoid Functions

Two-parameter evolution scheme means that the relationship between X_1 and X_2 is not linear anymore. Therefore, our one dimensional search landscape becomes two dimensional, where a one-parameter evolution represents a straight line in the two dimensional search space. The Hamiltonian can be written as:

$$H(s) = X_1(s) H_i + X_2(s) H_f, \tag{4.6}$$

where $X_1(0) = 1$, $X_1(1) = 0$, $X_2(0) = 0$ and $X_2(1) = 1$.

In this section, we are going to optimise two sigmoid functions, where for X_1 and X_2 in the Hamiltonian of Eq. (4.6), we put $X_1 = 1 - sig_{A,B}(s)$ and $X_2 = sig_{C,D}(s)$. For the same factoring instances that we have been looking at so far, we found the optimal parameters of the two sigmoid functions, where the annealing time was minimised with respect to the 4 numbers A, B, C and D, at a fidelity of 0.9. The optimal sigmoid functions for the hardest problem of each size are plotted in Fig. 4.14. All the functions look alike, but 4×4 is different from the others.

We can gain insight into these results by plotting the two-dimensional gap landscape as a function of X_1 and X_2. That is, we can assume that each function is a dimension, this results in having a two dimensional gap space. In Fig. 4.15, we have plotted the Hamiltonian gaps as a function of X_1 and X_2. The linear path is equivalent to $X_2 = 1 - X_1$ and is plotted with a red line. This line goes through the dark area of the 2D gap plot, where there are smaller gap values. On the other hand the two optimal sigmoid functions are plotted with the yellow curve. This optimal path avoids more of the dark areas. By using two functions, we are capable of annealing the system through bigger gap values. Our optimised functions go through bigger gap values, and at the same time adjust the speed with the gap.
We remember from Fig. 4.11, that there was a bump in the gap of 4×4. By looking at the 2D gap landscape of this problem, we can see that the red linear line crosses two dark areas, therefore it caused the gap function to have that bump in it. Even the yellow line of the two sigmoid optimal path of 4×4 is totally different from the other problems. This is all due to fact that the 2D gap space of this particular problem is

different from the others.

The energy gap along the path is plotted in Fig. 4.16. These plots are analogous to the previous gap plots, but just on a different path. By comparing these plots with the plots in Fig. 4.14, we see that the slope of X_1 and X_2 are small, when gap is small. Again the gap function of 4×4 along the path is different from the others, and it has extra peaks. For this case, the two sigmoid functions are insufficient, because

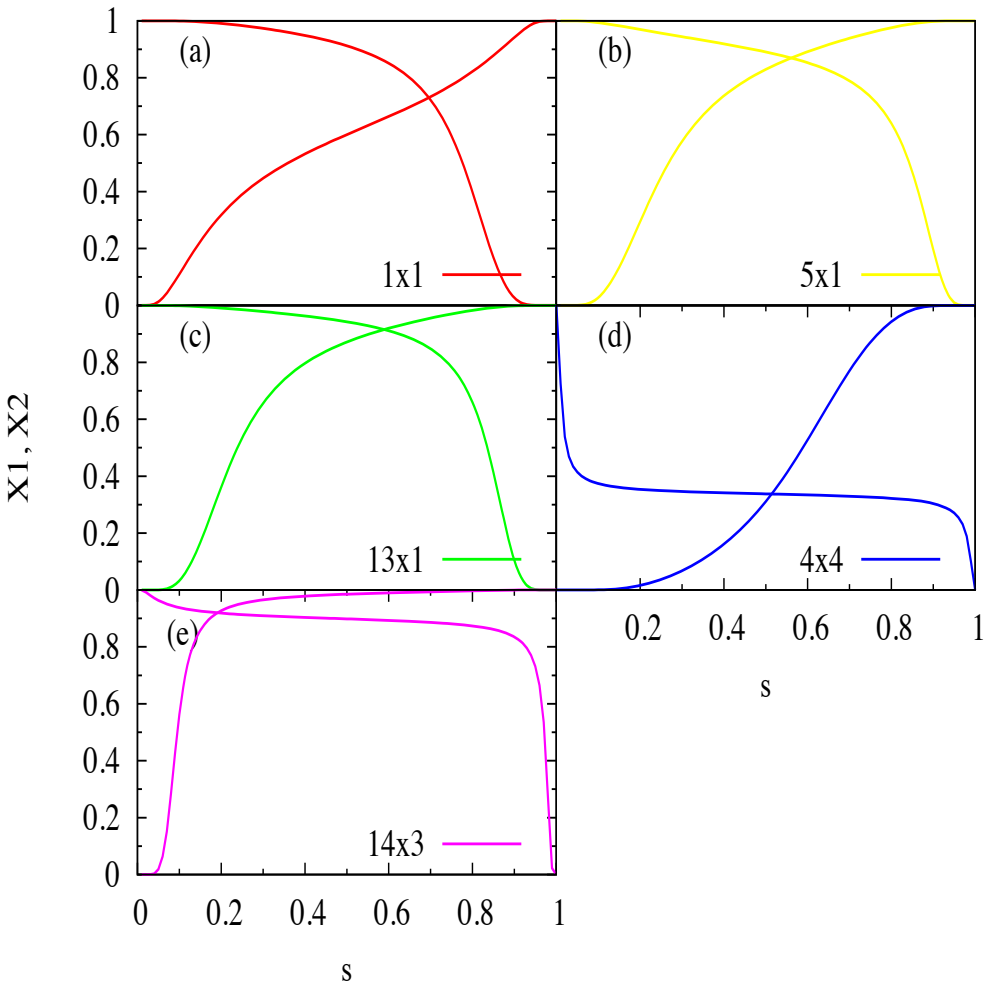

Figure 4.14: Optimal functions for a two-parameter evolution scheme, using two parametrised sigmoid functions. X_1 goes from 0 to 1, and X_2 goes from 1 to 0.

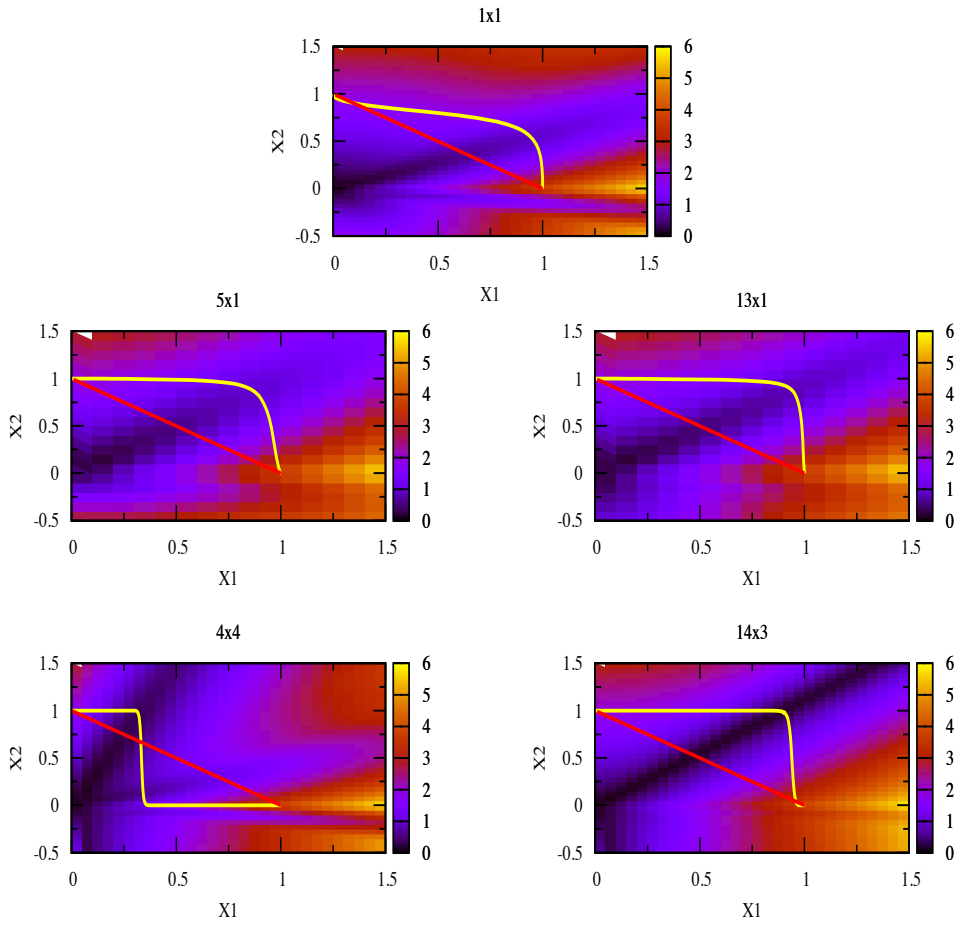

Figure 4.15: The two dimensional gap space, where the color represents the gap value. The red line represents the one-parameter evolution scheme, and the yellow curve is the path of the two optimal sigmoid functions for each problem in the two-parameter evolution scheme.

sigmoid functions can not adjust themselves with these extra peaks.

In Fig. 4.17, the plots of probability as a function of annealing time show that at almost all $\mathcal{T}$s the fidelity is greater when we optimised in the 2D landscape.

We find the minimum annealing times, after optimising the functions for the same problems that we have been looking at so far. These new annealing times are added to

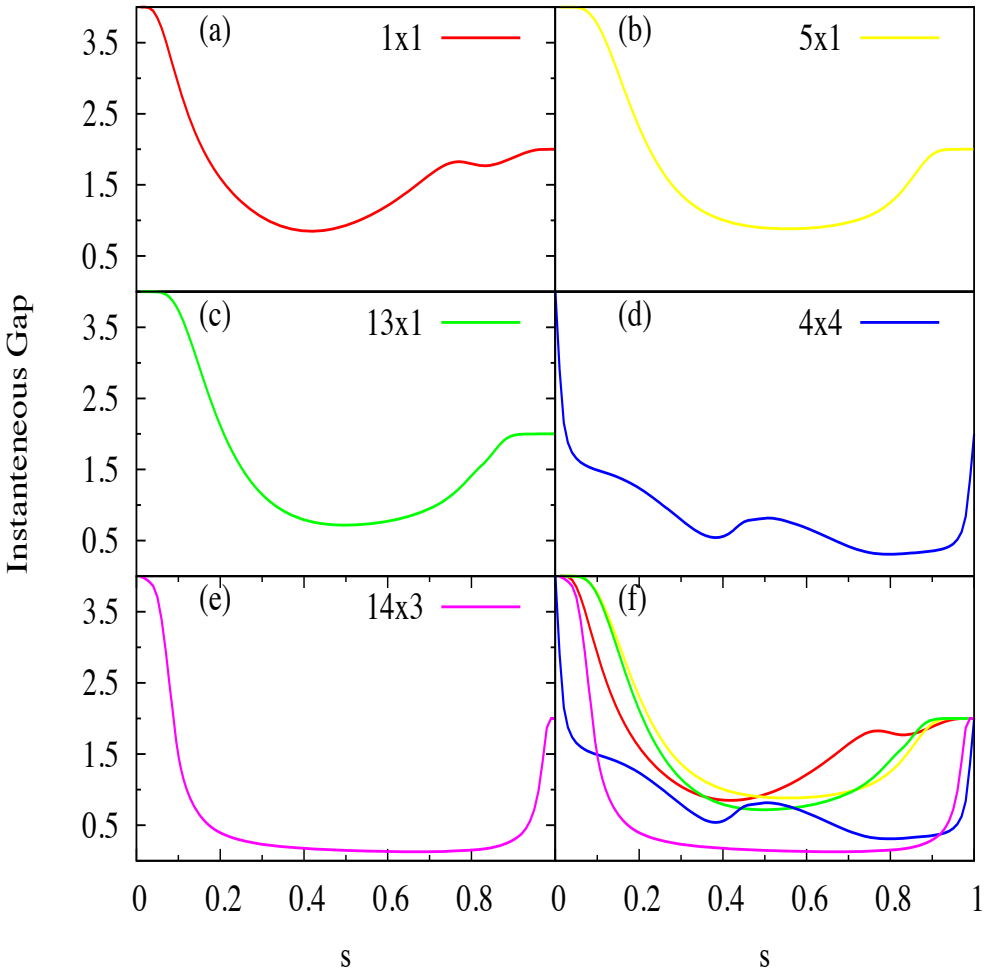

Figure 4.16: Instantaneous energy gap of Hamiltonians as a function of s, over the optimal path, which was derived by two sigmoid functions.

our list in Table 4.7. All annealing times for the fidelity of 0.9 are extremely reduced. This shows just switching from one dimension to two dimensions has great benefits. Although, we still do not know if the sigmoid functions are the appropriate functions to be optimised in two dimensions or not.

For a better comparison between the results of all the optimisation methods used so far, we have plotted the annealing times of the hardest problem of each size (the yellow colored boxes in the Table 4.7) versus problem size in Fig. 4.18. In the worst case

scenario of each size, using two sigmoid functions always results in shorter annealing times in comparison with other methods. In the previous sections, the approximation of points to exponential and polynomial functions was explained. We approximated the green points in the same manner, where the polynomial function fitted better than the exponential function. The approximated values of a, b, and c of the function $aN^b + c$ are listed in Table 4.8.

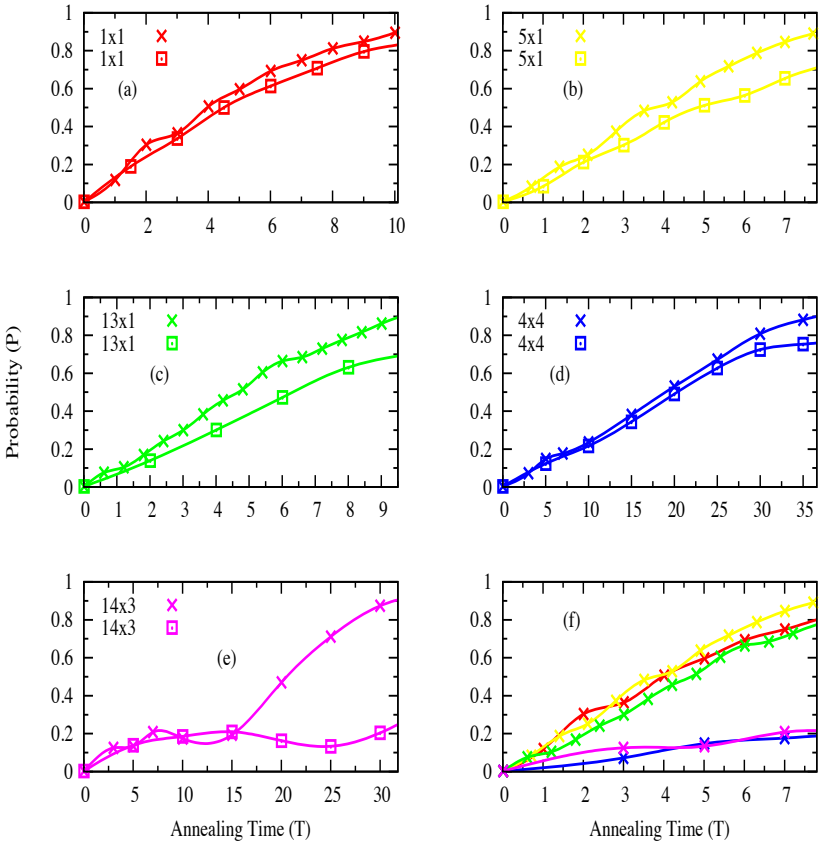

Figure 4.17: Probability of finding the ground state of the final Hamiltonian. The crossed lines show the probabilities using two sigmoid functions. The square lines show the probabilities using one sigmoid function.

The annealing time of the hardest problem of size five, which is 4×4, is bigger than the annealing time of the hardest problem of size six, which is 14×3. This makes the function approximation a bit difficult, and the big uncertainties in the approximation are for the same reason. As we discussed before, because of the distinct gap landscape of 4×4, the sigmoid function is not a suitable optimization function and consequently the optimised annealing path is not good for this problem. Therefore, the annealing time is long.

4.2.2 Two Hump Functions

Here I argue that we can look at the 2D gap to find better paths just by knowing that having a path that goes through bigger gaps gives a shorter annealing time. If we look at all 2D gaps we can see all the plots are similar, and by tracing a butterfly wing shape one can avoid very small gaps. This way, we suggest that using two

Size	Circuit	Number	Linear	EL	Sigmoid	2×Sigmoid
1 Bit	3 by 3 bits	1 = 1 x 1	26.018	18.4	12.435	10.147
3 Bit	4 by 2 bits	5 = 5 x 1	24.859	27.4	11.934	7.834
	4 by 2 bits	7 = 7 x 1	19.601	27.8	12.082	7.500
4 Bit	3 by 3 bits	9 = 3 x 3	27.096	44.9	12.584	7.782
	4 by 2 bits	11 = 11 x 1	38.447	32.1	15.604	8.420
	4 by 2 bits	13 = 13 x 1	47.960	35	17.131	9.573
5 Bit	3 by 3 bits	16 = 4 x 4	116.383	47.4	51.427	36.643
	4 by 2 bits	20 = 10 x 2	30.931	19.5	15.075	11.478
	4 by 2 bits	21 = 7 x 3	215	90.3	51.011	18.017
	4 by 2 bits	22 = 11 x 2	15.856	22.1	10.08	7.720
	3 by 3 bits	25 = 5 x 5	168.264	43.0	27.180	14.452
	4 by 2 bits	26 = 13 x 2	16.325	19.1	9.721	7.473
	4 by 2 bits	27 = 9 x 3	34.772	27.1	15.101	7.809
	4 by 2 bits	28 = 14 x 2	11.462	11.8	9.302	7.034
6 Bit	4 by 4 bits	33 = 11 x 3	27.803	12.1	12.022	6.752
	4 by 2 bits	39 = 13 x 3	65.420	23.4	18.319	8.231
	4 by 2 bits	42 = 14 x 3	1885.5	150.8	101.007	31.870
	3 by 3 bits	49 = 7 x 7	31.728	11.8	9.714	7.710

Table 4.7: Annealing times for fidelity of 0.9 using two sigmoid functions. The largest annealing time is highlighted in each bit sector.

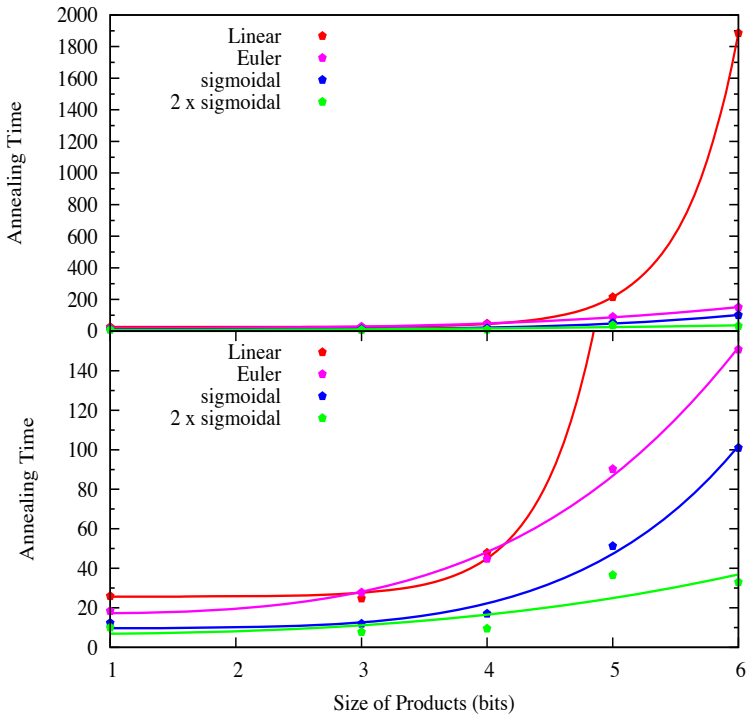

Figure 4.18: Scaling of the computation time with the problem size. Red points: linear function in the one-parameter evolution scheme. Violet points: EL function in the one-parameter evolution scheme. Blue points: sigmoid function in the one-parameter evolution scheme. Green points: two sigmoid functions in the two-parameter evolution scheme. All points were fit to a power law, except for the red curve, which is much better approximated by an exponential.

rms of residuals : 10.5373
variance of residuals : 111.036

Final set of parameters	Asymptotic standard error
$a = 0.215425$	$\pm 1.173\ (544.4\%)$
$b = 2.75864$	$\pm 2.957\ (107.2\%)$
$c = 6.67011$	$\pm 10.73\ \ (160.9\%)$

Table 4.8: Parameter values for fitting the worst annealing time points to a polynomial function, where two sigmoid functions in the two-parameter evolution scheme were used.

parametrised "hump" functions could be a better choice. Finally, we present the results of this path and compare it to the previous ones.

In the previous section, we traversed the two dimensional landscape using two sigmoid functions, and the annealing time was reduced compared to linear trajectories. In this section, we are going to use heuristics again, and derive another parametrised function, which is more suitable for our 2D landscapes. For reference, the plots of all 2D gaps for the problems that we have been dealing with are shown at the end, in Fig. 4.24, Fig. 4.25, Fig. 4.26.

All the gap landscapes have a butterfly shape. So we could make a function that can be adjusted with the gap shapes. In order to do that, we must be careful about three things. First, we have to find the paths where the gap values are bigger. These paths are found on the edge of the butterfly wing. Second, we must adjust the speed along the path, where in case of smaller gaps we need lower speeds. And third, we should have vanishing slopes at the beginning and the end of evolution. Therefore, we must derive two functions that have a hump shape, and satisfy the other needed conditions. We consider the following two functions that also satisfy the required boundary conditions:

$$X_1 = \frac{(s-1)^2 \left(3s^2 - 2s - 1 + 2A\,(2s+1)\right)}{2A-1} + B\,(s-1)^4\, s^2$$

$$X_2 = \frac{3s^4 - 4s^3\,(C+1) + 6Cs^2}{2C-1} + Ds^2\,(s-1)^2 \tag{4.7a}$$

The functions X_1 and X_2 are placed in the Hamiltonian of Eq. (4.6). These are the two new functions, which we call hump functions. For each problem, there are four parameters to be optimised. For the same factoring instances that we have been looking at so far, we found the optimal parameters of the two hump functions, where the annealing time was minimised at a fidelity of 0.9. The optimal hump functions for the most difficult problem of each size are plotted in Fig. 4.19. Note that these functions are in principal unbounded; only their end points are fixed.

In Fig. 4.20, we have plotted the optimal paths in the 2D gap landscape for the most difficult problem of each size. The range that the sigmoid function could cover was $0 \leq X_i \leq 1$, where $i = 1, 2$. The hump functions provide a compromise between path length and speed. Beyond a certain path length, the increase in the size of

the gap (and hence the allowable speed) is not sufficient to overcome the increased distance which must be traversed. In these plots we see that the optimal hump paths account for these requirements so nicely.

In Fig. 4.21, the gap functions along the optimal hump paths are plotted. These gaps are bigger than the gaps of the other paths. Specially the gap along the optimal path of 1×1 is much bigger than what it was before. Although the gap functions have more peaks now, the two hump functions have adjusted peaks so that the speed is matched with the gap value.

Size	Circuit	Number	Linear	EL	Sig	2×Sig	2×Hump
1 Bit	3 by 3 bits	1 = 1 x 1	26.018	18.4	12.435	10.147	2.082
3 Bit	4 by 2 bits	5 = 5 x 1	24.859	27.4	11.934	7.834	4.837
	4 by 2 bits	7 = 7 x 1	19.601	27.8	12.082	7.500	1.274
4 Bit	3 by 3 bits	9 = 3 x 3	27.096	44.9	12.584	7.782	5.828
	4 by 2 bits	11 = 11 x 1	38.447	32.1	15.604	8.420	6.539
	4 by 2 bits	13 = 13 x 1	47.960	35	17.131	9.573	6.114
5 Bit	3 by 3 bits	16 = 4 x 4	116.383	47.4	51.427	36.643	21.069
	4 by 2 bits	20 = 10 x 2	30.931	19.5	15.075	11.478	5.885
	4 by 2 bits	21 = 7 x 3	215	90.3	51.011	18.017	9.931
	4 by 2 bits	22 = 11 x 2	15.856	22.1	10.08	7.720	0.748
	3 by 3 bits	25 = 5 x 5	168.264	43.0	27.180	14.452	7.416
	4 by 2 bits	26 = 13 x 2	16.325	19.1	9.721	7.473	1.185
	4 by 2 bits	27 = 9 x 3	34.772	27.1	15.101	7.809	5.396
	4 by 2 bits	28 = 14 x 2	11.462	11.8	9.302	7.034	0.385
6 Bit	4 by 4 bits	33 = 11 x 3	27.803	12.1	12.022	6.752	3.681
	4 by 2 bits	39 = 13 x 3	65.420	23.4	18.319	8.231	6.578
	4 by 2 bits	42 = 14 x 3	1885.5	150.8	101.007	31.870	34.921
	3 by 3 bits	49 = 7 x 7	31.728	11.8	9.714	7.710	3.267

Table 4.9: Annealing times for fidelity of 0.9 using two hump functions. The largest annealing time is highlighted in each bit sector.

In Fig. 4.22, the plots of probability as a function of annealing time show that at almost all $\mathcal{T}$s the probability of finding the answer is greater, when the two optimal hump functions are used in the evolution.

We find the minimum annealing times, after optimising functions for the same problems that we have been looking at so far. These new annealing times are added to our list in Table 4.9. All annealing times for the fidelity of 0.9 are extremely reduced. This shows that using two sigmoid functions in 2D landscape is not the best option,

rms of residuals : 2.50451
variance of residuals :6.27257

Final set of parameters	Asymptotic standard error
$a = 0.0429175$	$\pm 0.06522\,(152\%)$
$b = 3.72259$	$\pm 0.8334\ (22.39\%)$
$c = 1.64322$	$\pm 2.229\ \ (135.7\%)$

Table 4.10: Parameter values for approximating the worst annealing time points with a polynomial function, where two hump functions were used in the evolution.

and there exists better functions like hump function.

For a better comparison between the results of all the optimisation methods used so far, we have plotted in Fig. 4.23 the annealing times of the most difficult problem of each size (the yellow colored boxes in the Table 4.9) versus problem size. In the worst case scenario of each size, using two hump functions always results in shorter annealing times in comparison with other methods. In the previous sections the approximation of points to exponential and polynomial functions was explained. We approximated the yellow points in the same manner, where the polynomial function fitted better than the exponential function. The approximated values of a, b, and c of the function $aN^b + c$ are listed in Table 4.10.

In Table 4.11, the fitting parameters for all the annealing functions that were used are shown. Getting an actual scaling law is very difficult because we know that the time can vary a lot depending on the annealing path. However we can say that the fitting is not exponential for the first six bits.

Our results indicate that there always exists paths with larger gaps, such that the problem can be solved in shorter annealing time. We have preliminary evidence that in higher dimensions even better paths can be found. In order to utilize those dimensions their gap landscape must be viewed, and an appropriate parameterized function must be found for it. Having access to higher dimensions, allow us to acquire even more optimal annealing schedules.

The same procedure can be followed for solving problems other than integer factorization. Although we do not have enough evidence, but a generic gap landscape seems to be general for all problems. Therefore good heuristics might be the key to efficiently computing on adiabatic quantum computers.

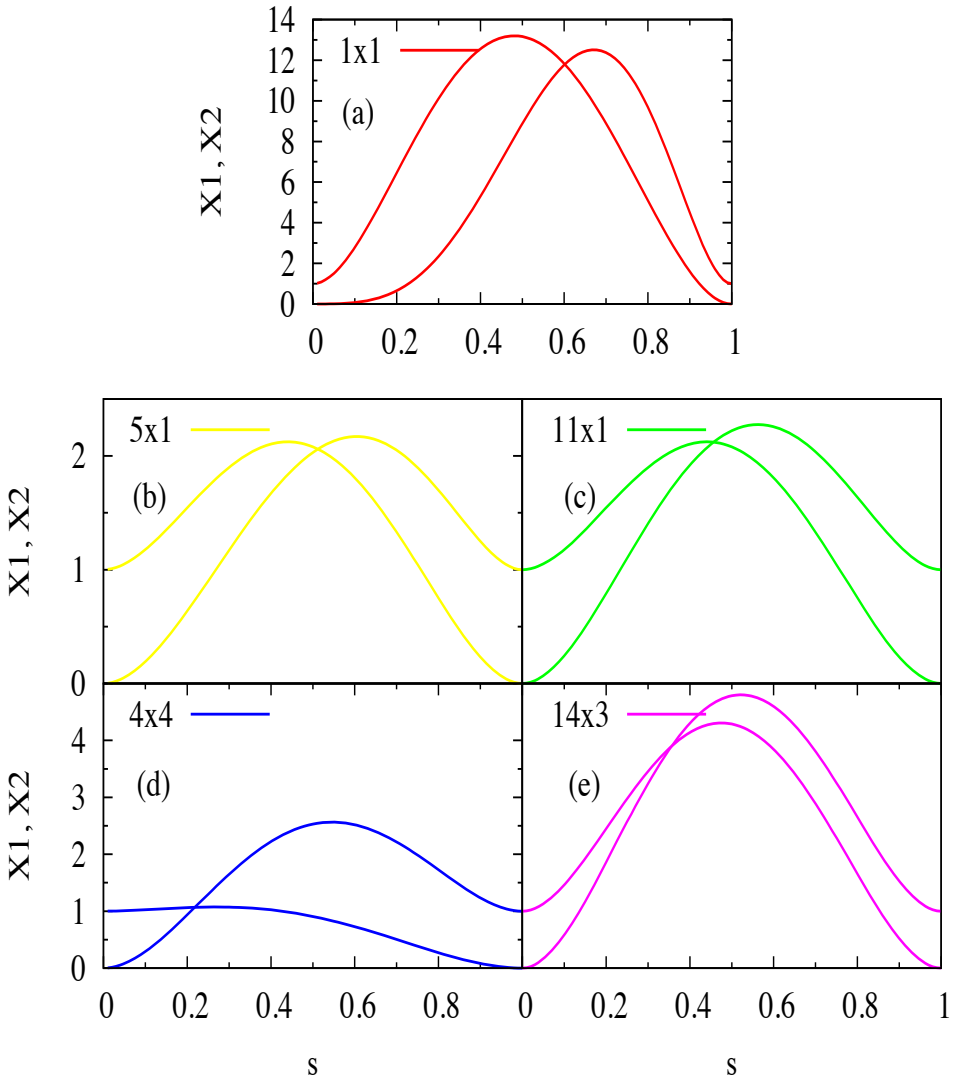

Figure 4.19: Optimal functions for the two-parameter scheme, using two parametrised hump functions. X_1 goes from 0 to 1, and X_2 goes from 1 to 0.

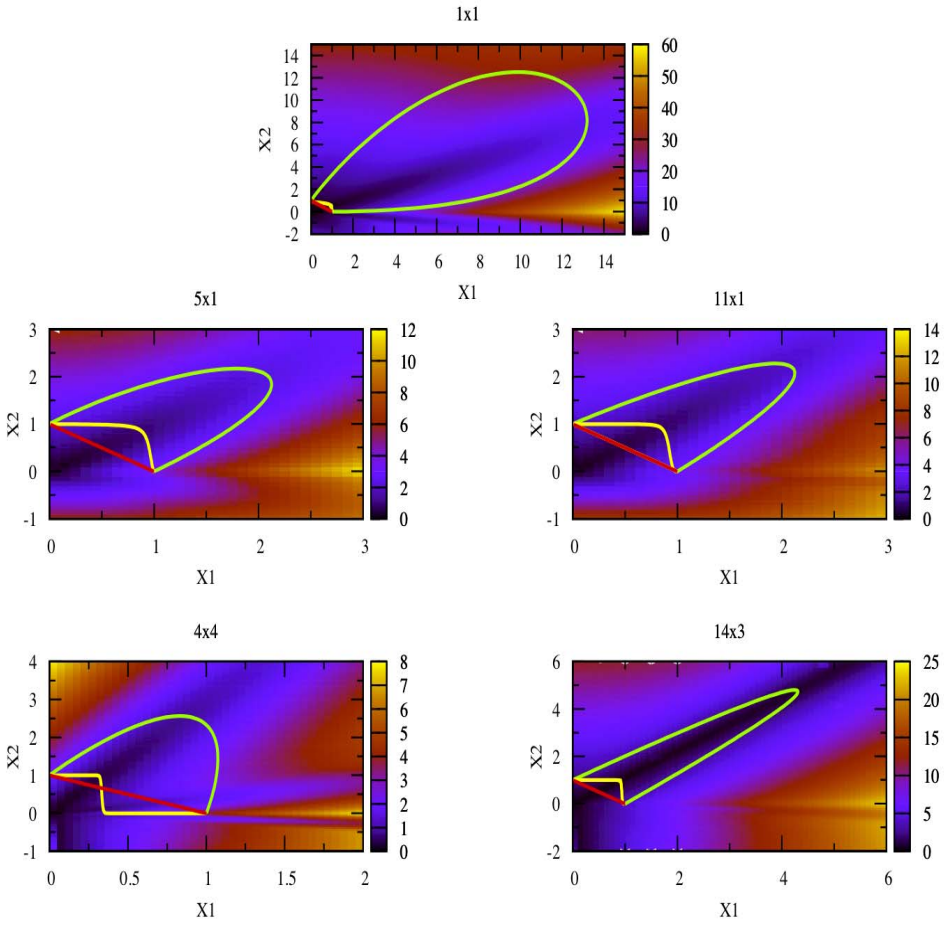

Figure 4.20: The two dimensional gap space, where the color represents the gap value. The red line is the linear path. The yellow curve is the path of the two optimal sigmoid functions. The green curve is the path of the two optimal hump functions.

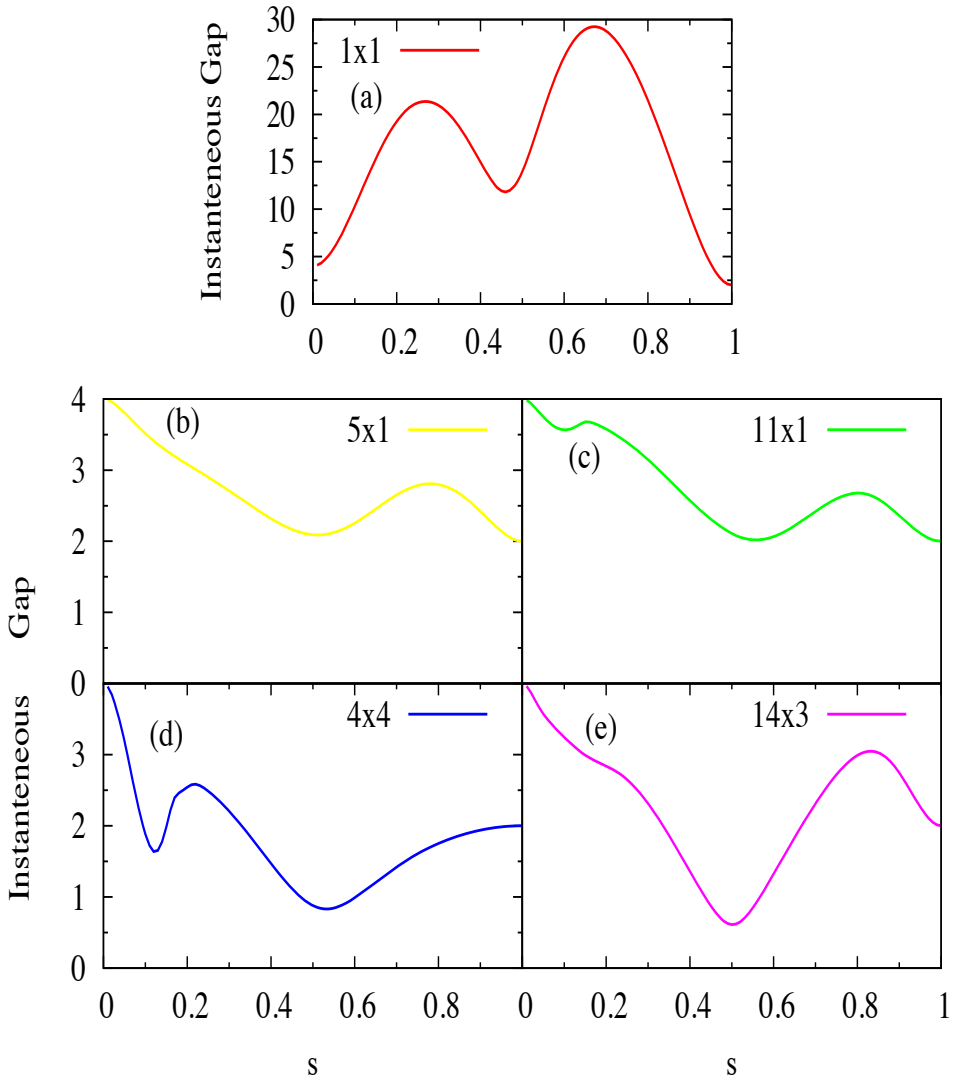

Figure 4.21: Instantaneous energy gap of Hamiltonians as a function of s, over the optimal path, which was derived by two hump functions.

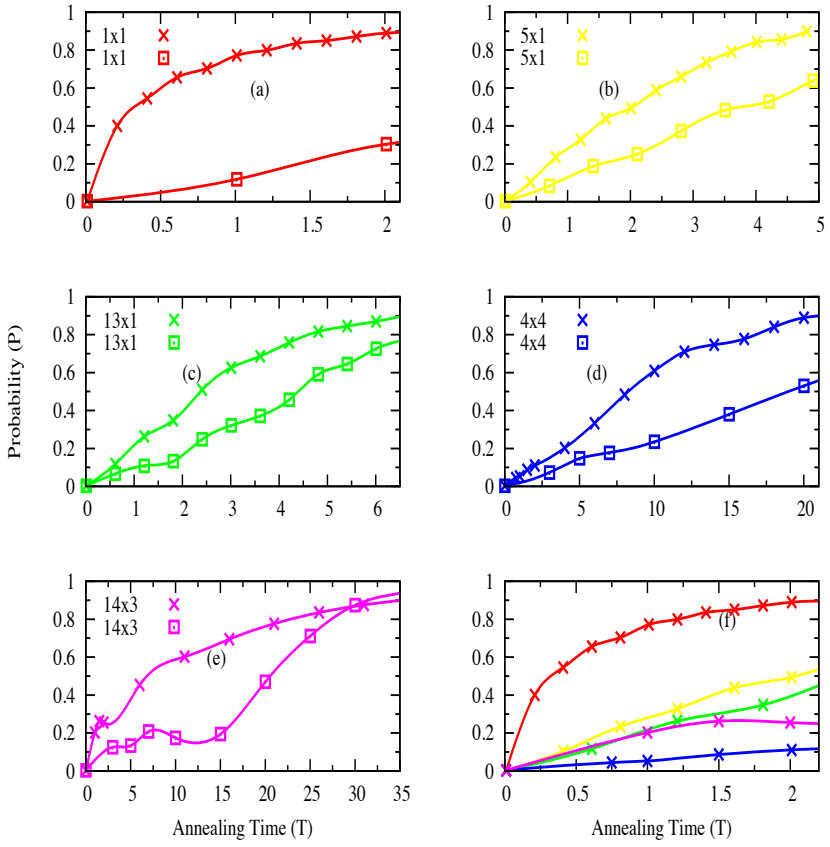

Figure 4.22: Probability of finding the ground state of the final Hamiltonian. The crossed lines show the probabilities using two hump functions. The square lines show the probabilities using two sigmoid function.

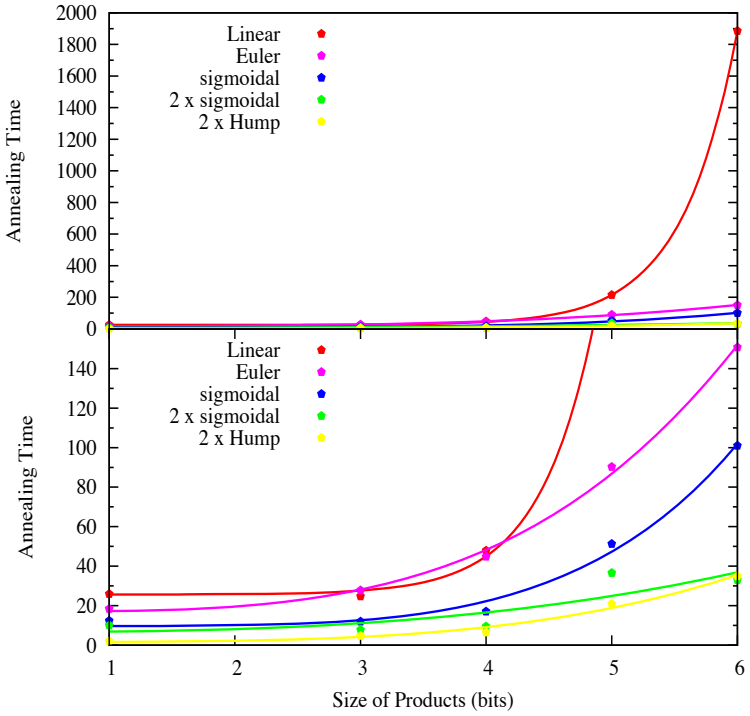

Figure 4.23: Scaling of the computation time with the problem size. Red points: using linear function in a linear path. Violet points: using EL function in a linear path. Blue points: using sigmoid function in a linear path. Green points: using two sigmoid functions. Yellow points: using two hump functions. All points were fit to a power law, except for the red curve, which is much better approximated by an exponential.

Method	Final set of parameters	Asymptotic standard error
Linear	$a = 0.00210702$	$\pm 0.000229\,(10.87\%)$
	$b = 2.28179$	$\pm 0.01792\ \ (0.7852\%)$
	$c = 25.6102$	$\pm 1.802\ \ \ (7.036\%)$
Euler	$a = 0.210153$	$\pm 0.1109\,(52.77\%)$
	$b = 3.60734$	$\pm 0.2892\ (8.017\%)$
	$c = 17.026$	$\pm 3.224\ \ (18.93\%)$
Sigmoid	$a = 0.0137642$	$\pm 0.02068\,(150.2\%)$
	$b = 4.91713$	$\pm 0.8276\ (16.83\%)$
	$c = 9.6529$	$\pm 4.004\ \ (41.47\%)$
2×Sigmoid	$a = 0.215425$	$\pm 1.173\,(544.4\%)$
	$b = 2.75864$	$\pm 2.957\ (107.2\%)$
	$c = 6.67011$	$\pm 10.73\ \ (160.9\%)$
2×Hump	$a = 0.0429175$	$\pm 0.06522\,(152\%)$
	$b = 3.72259$	$\pm 0.8334\ (22.39\%)$
	$c = 1.64322$	$\pm 2.229\ \ (135.7\%)$

Table 4.11: Fitting parameters for all of the annealing functions

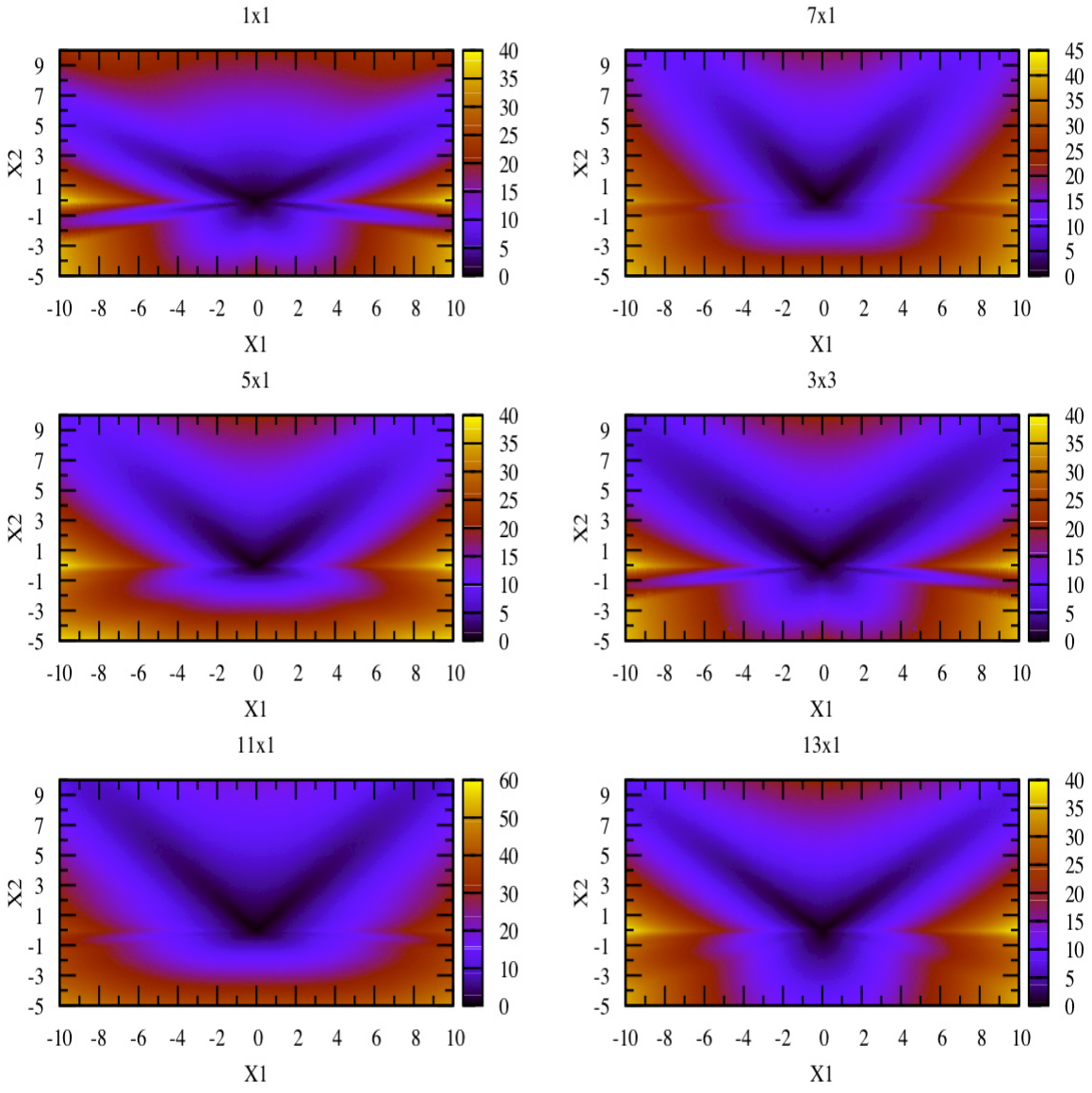

Figure 4.24: The two dimensional gap landscape.

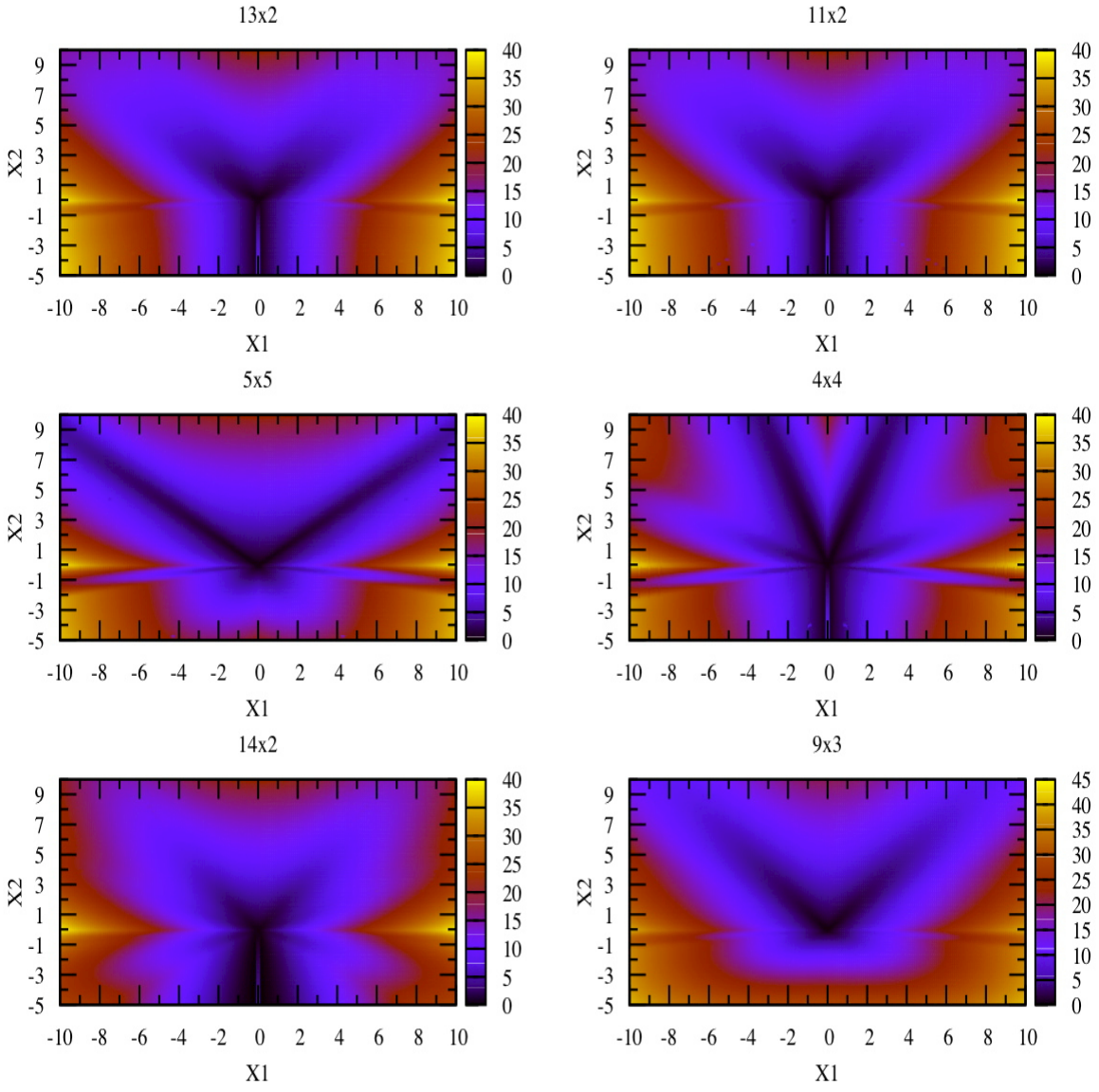

Figure 4.25: The two dimensional gap landscape.

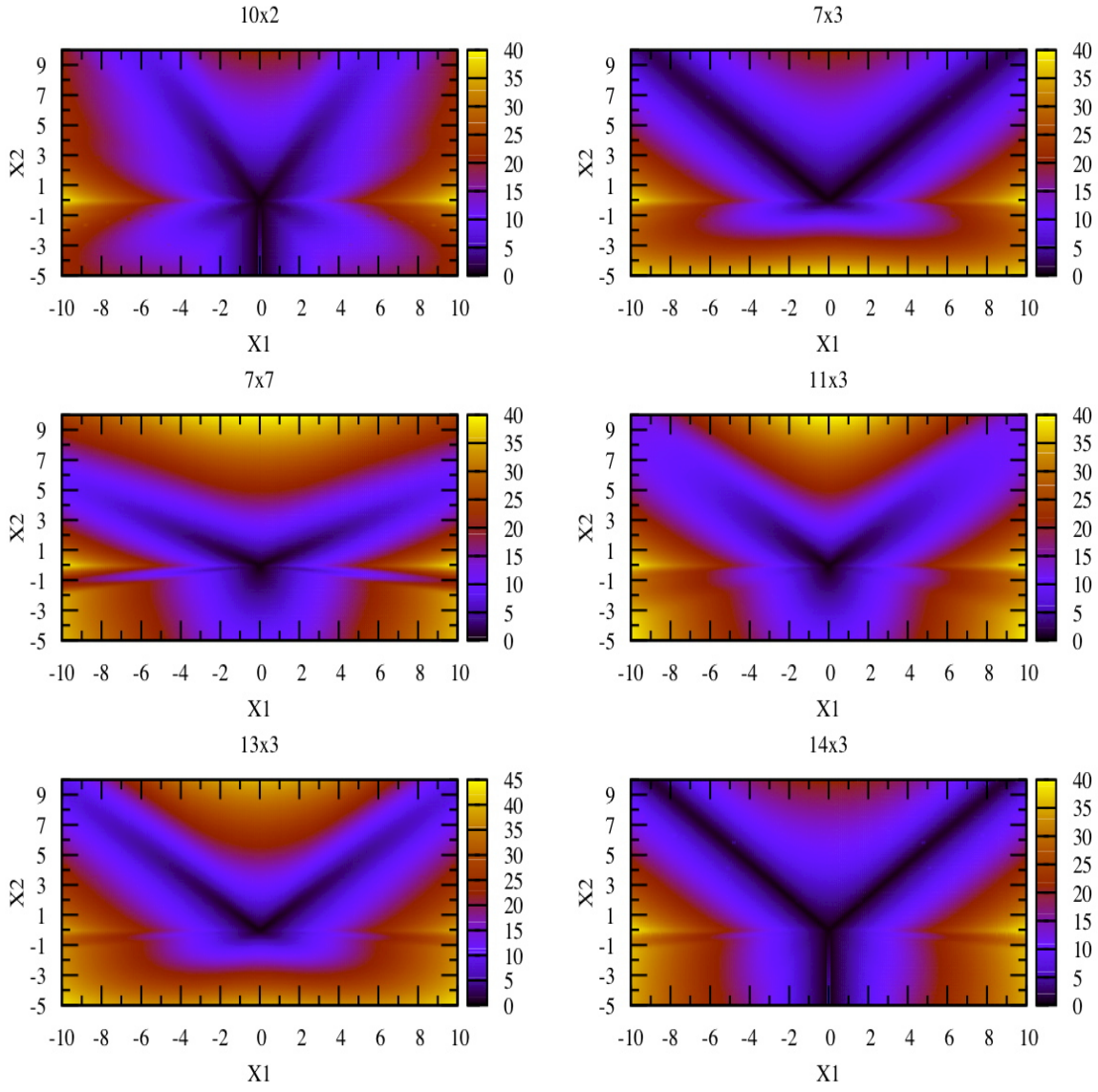

Figure 4.26: The two dimensional gap landscape.

Chapter 5

Conclusion

The computational power of circuit model quantum computers is dramatically illustrated via Shor's algorithm for integer factorization, where this quantum algorithm is exponentially more efficient than any known classical algorithm. On the other hand, there is no mathematical formulation for adiabatic quantum computers, such that one can assert any scaling between the problem size and computation time. However, through a limited number of factorization problems that could be solved numerically, we demonstrated such scaling. We showed that naive AQC is incapable of providing an exponential speedup over classical computers, but its unleashed power lies in the evolution path. Intuitive from the adiabatic theorem, the main ingredients of the AQC's speed are larger gaps and slower speeds. We brought conclusive evidence that there exist evolution paths that provide such ingredients. We provided one-parameter and two-parameter evolution schemes. Utilizing heuristically-derived parametrised functions, it was shown how these functions could find larger gaps, and could adjust their speed with the gap value to result in a faster computation. Not only could AQC factorize integers in polynomial time, but also the degree of the polynomial can be reduced by optimising the evolution path.

We have preliminary evidence that there exist even more optimal paths in higher dimensions, although only results for one-dimensional and two-dimensional landscapes were shown. We have developed a scheme to penetrate higher dimensions, in order to find short paths with larger gaps. The generic form of the gap functions for all the factorization problems allow an iterative process for finding the optimal parameters. Therefore, AQC has the potential to be not only faster than the classical computers, but also the circuit model quantum computer. These issues are the subject of ongoing work.

Appendix A

The Field and Coupling Terms for Solving 14×3

The matrix A, represents the field h_i and coupling c_{ij} terms of the spin glass Hamiltoninan Eq 3.1, for solving 14×3. $A_{ii} = h_i$ and $A_{ij} = c_{ij}$ for $i \neq j$.

$$A = \begin{pmatrix}
-4 & 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 \\
 & -4 & 0 & 0 & 1 & 1 & -2 & 2 & 0 & -2 & 0 & 0 & 0 & -4 & 0 & 0 & 0 \\
 & & -4 & 0 & 1 & 1 & 0 & -2 & 2 & 0 & -2 & 0 & 0 & 0 & -4 & 0 & 0 \\
 & & & -2 & 1 & 1 & 0 & 0 & -2 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\
 & & & & -2 & 0 & -2 & -2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & -12 & 2 & 2 & 2 & -2 & -2 & -2 & -4 & -4 & -4 & 0 & 0 \\
 & & & & & & -6 & 0 & 0 & 0 & 0 & 0 & -6 & 0 & 0 & 0 & 0 \\
 & & & & & & & -2 & 0 & -4 & 0 & 0 & 0 & -6 & 0 & 0 & 0 \\
 & & & & & & & & -2 & 0 & -4 & 0 & 0 & 0 & -6 & 0 & 0 \\
 & & & & & & & & & 4 & 0 & 0 & 1 & 6 & 0 & -2 & 0 \\
 & & & & & & & & & & 3 & 0 & 0 & 1 & 6 & 1 & -2 \\
 & & & & & & & & & & & 1 & 0 & 0 & 1 & 0 & 1 \\
 & & & & & & & & & & & & 14 & 0 & 0 & -2 & 0 \\
 & & & & & & & & & & & & & 7 & 0 & 1 & -2 \\
 & & & & & & & & & & & & & & 7 & 0 & 1 \\
 & & & & & & & & & & & & & & & -1 & -2 \\
 & & & & & & & & & & & & & & & & 1
\end{pmatrix}$$

Bibliography

[1] Shor P 1994 Algorithms for quantum computation: discrete logarithms and factoring *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on* pp 124 –134

[2] Zettili N 2001 *Quantum mechanics: concepts and applications* (John Wiley & Sons)

[3] Einstein A, Podolsky B and Rosen N 1935 *Phys. Rev.* **47** 777–780

[4] Bell J S 1964 *Physics* **1** 195–200

[5] Sakurai J J 1994 *Modern Quantum Mechanics* (Addison-Wesley)

[6] Townsend J S 2000 *A modern approach to Quantum Mechanics* (University Science Books)

[7] Dan C Marinescu G M M 2004 *Approaching Quantum Computing* (Pearson Education, Inc.)

[8] Ekert A and Jozsa R 1996 *Rev. Mod. Phys.* **68** 733–753

[9] Vidal G 2003 *Phys. Rev. Lett.* **91** 147902

[10] Michael A Nielsen I L C 2000 *Quantum Computation and Quantum Information* (Cambridge University Press)

[11] Ozawa M 1998 *Phys. Rev. Lett.* **80** 631–634

[12] DiVincenzo D 2000 *Fortschr. Phys* **48** 771–783

[13] Kok P, Munro W J, Nemoto K, Ralph T C, Dowling J P and Milburn G J 2007 *Rev. Mod. Phys.* **79** 135–174

[14] Knill E, Laflamme R and Milburn G J 2001 *Nature* **409** 46–52

[15] Hffner H, Roos C and Blatt R 2008 *Phys. Rep.* **469** 155 – 203

[16] Olmschenk S, Matsukevich D N, Maunz P, Hayes D, Duan L M and Monroe C 2009 *Science* **323** 486–489

[17] Morsch O and Oberthaler M 2006 *Rev. Mod. Phys.* **78** 179–215

[18] Cory D G, Fahmy A F and Havel T F 1997 *Proc. Natl. Acad. Sci. USA* **94** 1634–1639

[19] Loss D and DiVincenzo D P 1998 *Phys. Rev. A* **57** 120–126

[20] Hanson R, Kouwenhoven L P, Petta J R, Tarucha S and Vandersypen L M K 2007 *Rev. Mod. Phys.* **79** 1217–1265

[21] Ladd T D, Jelezko F, Laflamme R, Nakamura Y, Monroe C and O'Brien J L 2010 *Nature* **464** 45–53

[22] MacKenzie R, Morin-Duchesne A, Paquette H and Pinel J 2007 *Phys. Rev. A* **76** 044102

[23] Farhi E, Goldstone J, Gutmann S and Sipser M 2000 (*Preprint* `quant-ph/0001106`)

[24] Aharonov D, Dam W v, Kempe J, Landau Z, Lloyd S and Regev O 2005 Adiabatic quantum computation is equivalent to standard quantum computation (*Preprint* `quant-ph/0405098v2`)

[25] Monz T, Schindler P, Barreiro J T, Chwalla M, Nigg D, Coish W A, Harlander M, Hänsel W, Hennrich M and Blatt R 2011 *Phys. Rev. Lett.* **106** 130506

[26] Jordan S P, Farhi E and Shor P W 2006 *Phys. Rev. A* **74** 052322

[27] Johnson M W, Amin M H S, Gildert S, Lanting T, Hamze F, Dickson N, Harris R, Berkley A J, Johansson J, Bunyk P, Chapple E M, Enderud C, Hilton J P, Karimi K, Ladizinsky E, Ladizinsky N, Oh T, Perminov I, Rich C, Thom M C, Tolkacheva E, Truncik C J S, Uchaikin S, Wang J, Wilson B and Rose G 2011 *Nature* **473** 194–198

[28] Lenstra A K 2000 *Des. Code Cryptogr* **19** 101–128

[29] Jozsa R and Linden N 2003 *Proc. R. Soc. Lond. A* **459** 2011–2032

[30] Van Dam W, Mosca M and Vazirani U 2001 How powerful is adiabatic quantum computation? *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on* pp 279 – 287

[31] Avron J E, Fraas M, Graf G M and Grech P 2010 *Phys. Rev. A* **82** 040304

[32] Farhi E, Goldstone J, Gutmann S, Lapan J, Lundgren A and Preda D 2001 *Science* **292** 472–475

[33] Hogg T 2003 *Phys. Rev. A* **67** 022314

[34] Peierls R 1936 *Math. Proc. Cambridge* **32** 477–481

[35] Onsager L 1944 *Phys. Rev.* **65** 117–149

[36] Santoro G E, Martok R, Tosatti E and Car R 2002 *Science* **295** 2427–2430

[37] Barahona F 1985 *J. Phys. A* **18** L673

[38] Macready W, Drew-Brook M and Kyriakidis J 2009 *Factoring by Quantum Annealing* (Unpublished)

[39] Kato T 1950 *J. Phys. Soc. Japan* **5** 435–439

[40] Zener C 1932 *Proc. R. Soc. London, Ser. A* **137** 696–702

[41] Rezakhani A T, Pimachev A K and Lidar D A 2010 *Phys. Rev. A* **82** 052305

[42] Rezakhani A T, Kuo W J, Hamma A, Lidar D A and Zanardi P 2009 *Phys. Rev. Lett.* **103** 080502

[43] Roland J and Cerf N J 2002 *Phys. Rev. A* **65** 042308

[44] Morita S and Nishimori H 2008 *J. Math. Phys.* **49** 125210